Returns one of the Error code constants (@GlobalScope.OK on success).

Error **make_dir_absolute**(path: String) static 🔗

Static version of make_dir. Supports only absolute paths.

Error **make_dir_recursive**(path: String) 🔗

Creates a target directory and all necessary intermediate directories in its path, by calling make_dir recursively. The argument can be relative to the current directory, or an absolute path.

Returns one of the Error code constants (@GlobalScope.OK on success).

Error **make_dir_recursive_absolute**(path: String) static 🔗

Static version of make_dir_recursive. Supports only absolute paths.

DirAccess **open**(path: String) static 🔗

Creates a new **DirAccess** object and opens an existing directory of the filesystem. The `path` argument can be within the project tree (`res://folder`), the user directory (`user://folder`) or an absolute path of the user filesystem (e.g. `/tmp/folder` or `C:\tmp\folder`).

Returns `null` if opening the directory failed. You can use get_open_error to check the error that occurred.

String **read_link**(path: String) 🔗

Returns target of the symbolic link.

Note: This method is implemented on macOS, Linux, and Windows.

Error **remove**(path: String) 🔗

Permanently deletes the target file or an empty directory. The argument can be relative to the current directory, or an absolute path. If the target directory is not empty, the operation will fail.

If you don't want to delete the file/directory permanently, use OS.move_to_trash instead.

Returns one of the Error code constants (@GlobalScope.OK on success).

Error **remove_absolute**(path: String) static 🔗

Static version of remove. Supports only absolute paths.

Error **rename**(from: String, to: String) 🔗

Renames (move) the `from` file or directory to the `to` destination. Both arguments should be paths to files or directories, either relative or absolute. If the destination file or directory exists and is not access-protected, it will be overwritten.

Returns one of the Error code constants (@GlobalScope.OK on success).

Error **rename_absolute**(from: String, to: String) static 🔗

Static version of rename. Supports only absolute paths.

DisplayServer

Inherits: Object

A server interface for low-level window management.

Description

DisplayServer handles everything related to window management. It is separated from OS as a single operating system may support multiple display servers.

Headless mode: Starting the engine with the `--headless` command line argument disables all rendering and window management functions. Most functions from **DisplayServer** will return dummy values in this case.

Methods

String	clipboard_get() const
Image	clipboard_get_image() const
String	clipboard_get_primary() const
bool	clipboard_has() const
bool	clipboard_has_image() const
void	clipboard_set(clipboard: String)
void	clipboard_set_primary(clipboard_primary: String)
int	create_status_indicator(icon: Texture2D, tooltip: String, callback: Callable)

CursorShape	cursor_get_shape() const
void	cursor_set_custom_image(cursor: Resource, shape: CursorShape = 0, hotspot: Vector2 = Vector2(0, 0))
void	cursor_set_shape(shape: CursorShape)
void	delete_status_indicator(id: int)
Error	dialog_input_text(title: String, description: String, existing_text: String, callback: Callable)
Error	dialog_show(title: String, description: String, buttons: PackedStringArray, callback: Callable)
void	enable_for_stealing_focus(process_id: int)
Error	file_dialog_show(title: String, current_directory: String, filename: String, show_hidden: bool, mode: FileDialogMode, filters: PackedStringArray, callback: Callable)
Error	file_dialog_with_options_show(title: String, current_directory: String, root: String, filename: String, show_hidden: bool, mode: FileDialogMode, filters: PackedStringArray, options: Array[Dictionary], callback: Callable)
void	force_process_and_drop_events()
Color	get_accent_color() const

Color	get_base_color() const
Array[Rect2]	get_display_cutouts() const
Rect2i	get_display_safe_area() const
int	get_keyboard_focus_screen() const
String	get_name() const
int	get_primary_screen() const
int	get_screen_count() const
int	get_screen_from_rect(rect: Rect2) const
bool	get_swap_cancel_ok()
int	get_window_at_screen_position(position: Vector2i) const
PackedInt32Array	get_window_list() const
int	global_menu_add_check_item(menu_root: String, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	global_menu_add_icon_check_item(menu_root: String, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable =

	Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	global_menu_add_icon_item(menu_root: String, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	global_menu_add_icon_radio_check_item(menu_root: String, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	global_menu_add_item(menu_root: String, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	global_menu_add_multistate_item(menu_root: String, label: String, max_states: int, default_state: int, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	global_menu_add_radio_check_item(menu_root: String, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	global_menu_add_separator(menu_root: String, index: int = -1)
int	global_menu_add_submenu_item(menu_root: String, label: String, submenu: String, index: int = -1)
void	global_menu_clear(menu_root: String)

Key	global_menu_get_item_accelerator(menu_root: String, idx: int) const
Callable	global_menu_get_item_callback(menu_root: String, idx: int) const
int	global_menu_get_item_count(menu_root: String) const
Texture2D	global_menu_get_item_icon(menu_root: String, idx: int) const
int	global_menu_get_item_indentation_level(menu_root: String, idx: int) const
int	global_menu_get_item_index_from_tag(menu_root: String, tag: Variant) const
int	global_menu_get_item_index_from_text(menu_root: String, text: String) const
Callable	global_menu_get_item_key_callback(menu_root: String, idx: int) const
int	global_menu_get_item_max_states(menu_root: String, idx: int) const
int	global_menu_get_item_state(menu_root: String, idx: int) const
String	global_menu_get_item_submenu(menu_root: String, idx: int) const

Variant	global_menu_get_item_tag(menu_root: String, idx: int) const
String	global_menu_get_item_text(menu_root: String, idx: int) const
String	global_menu_get_item_tooltip(menu_root: String, idx: int) const
Dictionary	global_menu_get_system_menu_roots() const
bool	global_menu_is_item_checkable(menu_root: String, idx: int) const
bool	global_menu_is_item_checked(menu_root: String, idx: int) const
bool	global_menu_is_item_disabled(menu_root: String, idx: int) const
bool	global_menu_is_item_hidden(menu_root: String, idx: int) const
bool	global_menu_is_item_radio_checkable(menu_root: String, idx: int) const
void	global_menu_remove_item(menu_root: String, idx: int)
void	global_menu_set_item_accelerator(menu_root: String, idx: int, keycode: Key)

void	global_menu_set_item_callback(menu_root: String, idx: int, callback: Callable)
void	global_menu_set_item_checkable(menu_root: String, idx: int, checkable: bool)
void	global_menu_set_item_checked(menu_root: String, idx: int, checked: bool)
void	global_menu_set_item_disabled(menu_root: String, idx: int, disabled: bool)
void	global_menu_set_item_hidden(menu_root: String, idx: int, hidden: bool)
void	global_menu_set_item_hover_callbacks(menu_root: String, idx: int, callback: Callable)
void	global_menu_set_item_icon(menu_root: String, idx: int, icon: Texture2D)
void	global_menu_set_item_indentation_level(menu_root: String, idx: int, level: int)
void	global_menu_set_item_key_callback(menu_root: String, idx: int, key_callback: Callable)
void	global_menu_set_item_max_states(menu_root: String, idx: int, max_states: int)
void	global_menu_set_item_radio_checkable(menu_root: String, idx: int, checkable: bool)

void	global_menu_set_item_state(menu_root: String, idx: int, state: int)
void	global_menu_set_item_submenu(menu_root: String, idx: int, submenu: String)
void	global_menu_set_item_tag(menu_root: String, idx: int, tag: Variant)
void	global_menu_set_item_text(menu_root: String, idx: int, text: String)
void	global_menu_set_item_tooltip(menu_root: String, idx: int, tooltip: String)
void	global_menu_set_popup_callbacks(menu_root: String, open_callback: Callable, close_callback: Callable)
bool	has_additional_outputs() const
bool	has_feature(feature: Feature) const
void	help_set_search_callbacks(search_callback: Callable, action_callback: Callable)
Vector2i	ime_get_selection() const
String	ime_get_text() const
bool	is_dark_mode() const

bool	is_dark_mode_supported() const
bool	is_touchscreen_available() const
bool	is_window_transparency_available() const
int	keyboard_get_current_layout() const
Key	keyboard_get_keycode_from_physical(keycode: Key) const
Key	keyboard_get_label_from_physical(keycode: Key) const
int	keyboard_get_layout_count() const
String	keyboard_get_layout_language(index: int) const
String	keyboard_get_layout_name(index: int) const
void	keyboard_set_current_layout(index: int)
BitField[MouseButtonMask]	mouse_get_button_state() const
MouseMode	mouse_get_mode() const
Vector2i	mouse_get_position() const
void	mouse_set_mode(mouse_mode: MouseMode)
void	process_events()

void	register_additional_output(object: Object)
int	screen_get_dpi(screen: int = -1) const
Image	screen_get_image(screen: int = -1) const
float	screen_get_max_scale() const
ScreenOrientation	screen_get_orientation(screen: int = -1) const
Color	screen_get_pixel(position: Vector2i) const
Vector2i	screen_get_position(screen: int = -1) const
float	screen_get_refresh_rate(screen: int = -1) const
float	screen_get_scale(screen: int = -1) const
Vector2i	screen_get_size(screen: int = -1) const
Rect2i	screen_get_usable_rect(screen: int = -1) const
bool	screen_is_kept_on() const
void	screen_set_keep_on(enable: bool)
void	screen_set_orientation(orientation: ScreenOrientation, screen: int = -1)
void	set_icon(image: Image)

void	set_native_icon(filename: String)
void	set_system_theme_change_callback(callable: Callable)
Rect2	status_indicator_get_rect(id: int) const
void	status_indicator_set_callback(id: int, callback: Callable)
void	status_indicator_set_icon(id: int, icon: Texture2D)
void	status_indicator_set_menu(id: int, menu_rid: RID)
void	status_indicator_set_tooltip(id: int, tooltip: String)
String	tablet_get_current_driver() const
int	tablet_get_driver_count() const
String	tablet_get_driver_name(idx: int) const
void	tablet_set_current_driver(name: String)
Array[Dictionary]	tts_get_voices() const
PackedStringArray	tts_get_voices_for_language(language: String) const
bool	tts_is_paused() const
bool	tts_is_speaking() const

void	tts_pause()
void	tts_resume()
void	tts_set_utterance_callback(event: TTSUtteranceEvent, callable: Callable)
void	tts_speak(text: String, voice: String, volume: int = 50, pitch: float = 1.0, rate: float = 1.0, utterance_id: int = 0, interrupt: bool = false)
void	tts_stop()
void	unregister_additional_output(object: Object)
int	virtual_keyboard_get_height() const
void	virtual_keyboard_hide()
void	virtual_keyboard_show(existing_text: String, position: Rect2 = Rect2(0, 0, 0, 0), type: VirtualKeyboardType = 0, max_length: int = -1, cursor_start: int = -1, cursor_end: int = -1)
void	warp_mouse(position: Vector2i)
bool	window_can_draw(window_id: int = 0) const
int	window_get_active_popup() const
int	window_get_attached_instance_id(window_id: int = 0) const

int	window_get_current_screen(window_id: int = 0) const
bool	window_get_flag(flag: WindowFlags, window_id: int = 0) const
Vector2i	window_get_max_size(window_id: int = 0) const
Vector2i	window_get_min_size(window_id: int = 0) const
WindowMode	window_get_mode(window_id: int = 0) const
int	window_get_native_handle(handle_type: HandleType, window_id: int = 0) const
Rect2i	window_get_popup_safe_rect(window: int) const
Vector2i	window_get_position(window_id: int = 0) const
Vector2i	window_get_position_with_decorations(window_id: int = 0) const
Vector3i	window_get_safe_title_margins(window_id: int = 0) const
Vector2i	window_get_size(window_id: int = 0) const
Vector2i	window_get_size_with_decorations(window_id: int = 0) const
Vector2i	window_get_title_size(title: String, window_id: int = 0) const

VSyncMode	window_get_vsync_mode(window_id: int = 0) const
bool	window_is_focused(window_id: int = 0) const
bool	window_is_maximize_allowed(window_id: int = 0) const
bool	window_maximize_on_title_dbl_click() const
bool	window_minimize_on_title_dbl_click() const
void	window_move_to_foreground(window_id: int = 0)
void	window_request_attention(window_id: int = 0)
void	window_set_current_screen(screen: int, window_id: int = 0)
void	window_set_drop_files_callback(callback: Callable, window_id: int = 0)
void	window_set_exclusive(window_id: int, exclusive: bool)
void	window_set_flag(flag: WindowFlags, enabled: bool, window_id: int = 0)
void	window_set_ime_active(active: bool, window_id: int = 0)
void	window_set_ime_position(position: Vector2i, window_id: int = 0)

void	window_set_input_event_callback(callback: Callable, window_id: int = 0)
void	window_set_input_text_callback(callback: Callable, window_id: int = 0)
void	window_set_max_size(max_size: Vector2i, window_id: int = 0)
void	window_set_min_size(min_size: Vector2i, window_id: int = 0)
void	window_set_mode(mode: WindowMode, window_id: int = 0)
void	window_set_mouse_passthrough(region: PackedVector2Array, window_id: int = 0)
void	window_set_popup_safe_rect(window: int, rect: Rect2i)
void	window_set_position(position: Vector2i, window_id: int = 0)
void	window_set_rect_changed_callback(callback: Callable, window_id: int = 0)
void	window_set_size(size: Vector2i, window_id: int = 0)
void	window_set_title(title: String, window_id: int = 0)
void	window_set_transient(window_id: int, parent_window_id: int)

void	window_set_vsync_mode(vsync_mode: VSyncMode, window_id: int = 0)
void	window_set_window_buttons_offset(offset: Vector2i, window_id: int = 0)
void	window_set_window_event_callback(callback: Callable, window_id: int = 0)

Enumerations

enum **Feature**: 🔗

Feature **FEATURE_GLOBAL_MENU** = `0`

Deprecated: Use NativeMenu or PopupMenu instead.

Display server supports global menu. This allows the application to display its menu items in the operating system's top bar. **macOS**

Feature **FEATURE_SUBWINDOWS** = `1`

Display server supports multiple windows that can be moved outside of the main window. **Windows, macOS, Linux (X11)**

Feature **FEATURE_TOUCHSCREEN** = `2`

Display server supports touchscreen input. **Windows, Linux (X11), Android, iOS, Web**

Feature **FEATURE_MOUSE** = `3`

Display server supports mouse input. **Windows, macOS, Linux (X11/Wayland), Android, Web**

Feature **FEATURE_MOUSE_WARP** = `4`

Display server supports warping mouse coordinates to keep the mouse cursor constrained within an area, but looping when one of the edges is reached.

Windows, macOS, Linux (X11/Wayland)

Feature **FEATURE_CLIPBOARD** = `5`

Display server supports setting and getting clipboard data. See also FEATURE_CLIPBOARD_PRIMARY. **Windows, macOS, Linux (X11/Wayland), Android, iOS, Web**

Feature **FEATURE_VIRTUAL_KEYBOARD** = `6`

Display server supports popping up a virtual keyboard when requested to input text without a physical keyboard. **Android, iOS, Web**

Feature **FEATURE_CURSOR_SHAPE** = `7`

Display server supports setting the mouse cursor shape to be different from the default. **Windows, macOS, Linux (X11/Wayland), Android, Web**

Feature **FEATURE_CUSTOM_CURSOR_SHAPE** = `8`

Display server supports setting the mouse cursor shape to a custom image. **Windows, macOS, Linux (X11/Wayland), Web**

Feature **FEATURE_NATIVE_DIALOG** = `9`

Display server supports spawning text dialogs using the operating system's native look-and-feel. See dialog_show. **Windows, macOS**

Feature **FEATURE_IME** = `10`

Display server supports Input Method Editor [https://en.wikipedia.org/wiki/Input_method], which is commonly used for inputting Chinese/Japanese/Korean text. This is handled by the operating system, rather than by Godot. **Windows, macOS, Linux (X11)**

Feature **FEATURE_WINDOW_TRANSPARENCY** = `11`

Display server supports windows can use per-pixel transparency to make windows behind them partially or fully visible. **Windows, macOS, Linux (X11/Wayland)**

Feature **FEATURE_HIDPI** = `12`

Display server supports querying the operating system's display scale factor. This allows for *reliable* automatic hiDPI display detection, as opposed to guessing based on the screen resolution and reported display DPI (which can be unreliable due to broken monitor EDID). **Windows, Linux (Wayland), macOS**

Feature **FEATURE_ICON** = `13`

Display server supports changing the window icon (usually displayed in the top-left corner). **Windows, macOS, Linux (X11)**

Feature **FEATURE_NATIVE_ICON** = `14`

Display server supports changing the window icon (usually displayed in the top-left corner). **Windows, macOS**

Feature **FEATURE_ORIENTATION** = `15`

Display server supports changing the screen orientation. **Android, iOS**

Feature **FEATURE_SWAP_BUFFERS** = `16`

Display server supports V-Sync status can be changed from the default (which is forced to be enabled platforms not supporting this feature). **Windows, macOS, Linux (X11/Wayland)**

Feature **FEATURE_CLIPBOARD_PRIMARY** = `18`

Display server supports Primary clipboard can be used. This is a different clipboard from FEATURE_CLIPBOARD. **Linux (X11/Wayland)**

Feature **FEATURE_TEXT_TO_SPEECH** = `19`

Display server supports text-to-speech. See `tts_*` methods. **Windows, macOS, Linux (X11/Wayland), Android, iOS, Web**

Feature **FEATURE_EXTEND_TO_TITLE** = `20`

Display server supports expanding window content to the title. See WINDOW_FLAG_EXTEND_TO_TITLE. **macOS**

Feature **FEATURE_SCREEN_CAPTURE** = `21`

Display server supports reading screen pixels. See screen_get_pixel.

Feature **FEATURE_STATUS_INDICATOR** = `22`

Display server supports application status indicators.

Feature **FEATURE_NATIVE_HELP** = `23`

Display server supports native help system search callbacks. See help_set_search_callbacks.

Feature **FEATURE_NATIVE_DIALOG_INPUT** = `24`

Display server supports spawning text input dialogs using the operating system's native look-and-feel. See dialog_input_text. **Windows, macOS**

Feature **FEATURE_NATIVE_DIALOG_FILE** = `25`

Display server supports spawning dialogs for selecting files or directories using the operating system's native look-and-feel. See file_dialog_show and file_dialog_with_options_show. **Windows, macOS, Linux (X11/Wayland)**

---

enum **MouseMode**: 🔗

MouseMode **MOUSE_MODE_VISIBLE** = `0`

Makes the mouse cursor visible if it is hidden.

MouseMode **MOUSE_MODE_HIDDEN** = `1`

Makes the mouse cursor hidden if it is visible.

MouseMode **MOUSE_MODE_CAPTURED** = `2`

Captures the mouse. The mouse will be hidden and its position locked at the center of the window manager's window.

Note: If you want to process the mouse's movement in this mode, you need to use InputEventMouseMotion.relative.

MouseMode **MOUSE_MODE_CONFINED** = `3`

Confines the mouse cursor to the game window, and make it visible.

MouseMode **MOUSE_MODE_CONFINED_HIDDEN** = `4`

Confines the mouse cursor to the game window, and make it hidden.

---

enum **ScreenOrientation**: 🔗

ScreenOrientation **SCREEN_LANDSCAPE** = `0`

Default landscape orientation.

ScreenOrientation **SCREEN_PORTRAIT** = `1`

Default portrait orientation.

ScreenOrientation **SCREEN_REVERSE_LANDSCAPE** = `2`

Reverse landscape orientation (upside down).

ScreenOrientation **SCREEN_REVERSE_PORTRAIT** = `3`

Reverse portrait orientation (upside down).

ScreenOrientation **SCREEN_SENSOR_LANDSCAPE** = `4`

Automatic landscape orientation (default or reverse depending on sensor).

ScreenOrientation **SCREEN_SENSOR_PORTRAIT** = `5`

Automatic portrait orientation (default or reverse depending on sensor).

ScreenOrientation **SCREEN_SENSOR** = `6`

Automatic landscape or portrait orientation (default or reverse depending on sensor).

---

enum **VirtualKeyboardType**: 🔗

VirtualKeyboardType **KEYBOARD_TYPE_DEFAULT** = `0`

Default text virtual keyboard.

VirtualKeyboardType **KEYBOARD_TYPE_MULTILINE** = `1`

Multiline virtual keyboard.

VirtualKeyboardType **KEYBOARD_TYPE_NUMBER** = `2`

Virtual number keypad, useful for PIN entry.

VirtualKeyboardType **KEYBOARD_TYPE_NUMBER_DECIMAL** = `3`

Virtual number keypad, useful for entering fractional numbers.

VirtualKeyboardType **KEYBOARD_TYPE_PHONE** = `4`

Virtual phone number keypad.

VirtualKeyboardType **KEYBOARD_TYPE_EMAIL_ADDRESS** = `5`

Virtual keyboard with additional keys to assist with typing email addresses.

VirtualKeyboardType **KEYBOARD_TYPE_PASSWORD** = `6`

Virtual keyboard for entering a password. On most platforms, this should disable autocomplete and autocapitalization.

Note: This is not supported on Web. Instead, this behaves identically to KEYBOARD_TYPE_DEFAULT.

VirtualKeyboardType **KEYBOARD_TYPE_URL** = `7`

Virtual keyboard with additional keys to assist with typing URLs.

---

enum **CursorShape**: 🔗

CursorShape **CURSOR_ARROW** = `0`

Arrow cursor shape. This is the default when not pointing anything that overrides the mouse cursor, such as a LineEdit or TextEdit.

CursorShape **CURSOR_IBEAM** = `1`

I-beam cursor shape. This is used by default when hovering a control that accepts text input, such as LineEdit or TextEdit.

CursorShape **CURSOR_POINTING_HAND** = `2`

Pointing hand cursor shape. This is used by default when hovering a LinkButton or a URL tag in a RichTextLabel.

CursorShape **CURSOR_CROSS** = `3`

Crosshair cursor. This is intended to be displayed when the user needs precise aim over an element, such as a rectangle selection tool or a color picker.

CursorShape **CURSOR_WAIT** = `4`

Wait cursor. On most cursor themes, this displays a spinning icon *besides* the arrow. Intended to be used for non-blocking operations (when the user can do something else at the moment). See also CURSOR_BUSY.

CursorShape **CURSOR_BUSY** = `5`

Wait cursor. On most cursor themes, this *replaces* the arrow with a spinning icon. Intended to be used for blocking operations (when the user can't do anything else at the moment). See also CURSOR_WAIT.

CursorShape **CURSOR_DRAG** = `6`

Dragging hand cursor. This is displayed during drag-and-drop operations. See also CURSOR_CAN_DROP.

CursorShape **CURSOR_CAN_DROP** = `7`

"Can drop" cursor. This is displayed during drag-and-drop operations if hovering over a Control that can accept the drag-and-drop event. On most cursor themes, this displays a dragging hand with an arrow symbol besides it. See also CURSOR_DRAG.

CursorShape **CURSOR_FORBIDDEN** = `8`

Forbidden cursor. This is displayed during drag-and-drop operations if the hovered Control can't accept the drag-and-drop event.

CursorShape **CURSOR_VSIZE** = `9`

Vertical resize cursor. Intended to be displayed when the hovered Control can be vertically resized using the mouse. See also CURSOR_VSPLIT.

CursorShape **CURSOR_HSIZE** = `10`

Horizontal resize cursor. Intended to be displayed when the hovered Control can be horizontally resized using the mouse. See also CURSOR_HSPLIT.

CursorShape **CURSOR_BDIAGSIZE** = `11`

Secondary diagonal resize cursor (top-right/bottom-left). Intended to be displayed when the hovered Control can be resized on both axes at once using the mouse.

CursorShape **CURSOR_FDIAGSIZE** = `12`

Main diagonal resize cursor (top-left/bottom-right). Intended to be displayed when the hovered Control can be resized on both axes at once using the mouse.

CursorShape **CURSOR_MOVE** = `13`

Move cursor. Intended to be displayed when the hovered Control can be moved using the mouse.

CursorShape **CURSOR_VSPLIT** = `14`

Vertical split cursor. This is displayed when hovering a Control with splits that can be vertically resized using the mouse, such as VSplitContainer. On some cursor themes, this cursor may have the same appearance as CURSOR_VSIZE.

CursorShape **CURSOR_HSPLIT** = `15`

Horizontal split cursor. This is displayed when hovering a Control with splits that can be horizontally resized using the mouse, such as HSplitContainer. On some cursor themes, this cursor may have the same appearance as CURSOR_HSIZE.

CursorShape **CURSOR_HELP** = `16`

Help cursor. On most cursor themes, this displays a question mark icon instead of the mouse cursor. Intended to be used when the user has requested help on the next element that will be clicked.

CursorShape **CURSOR_MAX** = `17`

Represents the size of the CursorShape enum.

---

enum **FileDialogMode**: 🔗

FileDialogMode **FILE_DIALOG_MODE_OPEN_FILE** = `0`

The native file dialog allows selecting one, and only one file.

FileDialogMode **FILE_DIALOG_MODE_OPEN_FILES** = `1`

The native file dialog allows selecting multiple files.

FileDialogMode **FILE_DIALOG_MODE_OPEN_DIR** = `2`

The native file dialog only allows selecting a directory, disallowing the selection of any file.

FileDialogMode **FILE_DIALOG_MODE_OPEN_ANY** = `3`

The native file dialog allows selecting one file or directory.

FileDialogMode **FILE_DIALOG_MODE_SAVE_FILE** = `4`

The native file dialog will warn when a file exists.

---

enum **WindowMode**: 🔗

WindowMode **WINDOW_MODE_WINDOWED** = `0`

Windowed mode, i.e. Window doesn't occupy the whole screen (unless set to the size of the screen).

WindowMode **WINDOW_MODE_MINIMIZED** = `1`

Minimized window mode, i.e. Window is not visible and available on window manager's window list. Normally happens when the minimize button is pressed.

WindowMode **WINDOW_MODE_MAXIMIZED** = `2`

Maximized window mode, i.e. Window will occupy whole screen area except task bar and still display its borders. Normally happens when the maximize button is pressed.

WindowMode **WINDOW_MODE_FULLSCREEN** = `3`

Full screen mode with full multi-window support.

Full screen window covers the entire display area of a screen and has no decorations. The display's video mode is not changed.

On Windows: Multi-window full-screen mode has a 1px border of the ProjectSettings.rendering/environment/defaults/default_clear_color color.

On macOS: A new desktop is used to display the running project.

Note: Regardless of the platform, enabling full screen will change the window size to match the monitor's size. Therefore, make sure your project supports multiple resolutions when enabling full screen mode.

WindowMode **WINDOW_MODE_EXCLUSIVE_FULLSCREEN** = `4`

A single window full screen mode. This mode has less overhead, but only one window can be open on a given screen at a time (opening a child window or application switching will trigger a full screen transition).

Full screen window covers the entire display area of a screen and has no border or decorations. The display's video mode is not changed.

On Windows: Depending on video driver, full screen transition might cause screens to go black for a moment.

On macOS: A new desktop is used to display the running project. Exclusive full screen mode prevents Dock and Menu from showing up when the mouse pointer is hovering the edge of the screen.

On Linux (X11): Exclusive full screen mode bypasses compositor.

Note: Regardless of the platform, enabling full screen will change the window size to match the monitor's size. Therefore, make sure your project supports multiple resolutions when enabling full screen mode.

---

enum **WindowFlags**: 🔗

WindowFlags **WINDOW_FLAG_RESIZE_DISABLED** = `0`

The window can't be resized by dragging its resize grip. It's still possible to resize the window using window_set_size. This flag is ignored for full screen windows.

WindowFlags **WINDOW_FLAG_BORDERLESS** = `1`

The window do not have native title bar and other decorations. This flag is ignored for full-screen windows.

WindowFlags **WINDOW_FLAG_ALWAYS_ON_TOP** = `2`

The window is floating on top of all other windows. This flag is ignored for full-screen windows.

WindowFlags **WINDOW_FLAG_TRANSPARENT** = `3`

The window background can be transparent.

Note: This flag has no effect if is_window_transparency_available returns `false`.

Note: Transparency support is implemented on Linux (X11/Wayland), macOS, and Windows, but availability might vary depending on GPU driver, display manager, and compositor capabilities.

WindowFlags **WINDOW_FLAG_NO_FOCUS** = `4`

The window can't be focused. No-focus window will ignore all input, except mouse clicks.

WindowFlags **WINDOW_FLAG_POPUP** = `5`

Window is part of menu or OptionButton dropdown. This flag can't be changed when the window is visible. An active popup window will exclusively receive all input, without stealing focus from its parent. Popup windows are automatically closed when uses click outside it, or when an application is switched. Popup window must have transient parent set (see window_set_transient).

WindowFlags **WINDOW_FLAG_EXTEND_TO_TITLE** = `6`

Window content is expanded to the full size of the window. Unlike borderless window, the frame is left intact and can be used to resize the window, title bar is transparent, but have minimize/maximize/close buttons.

Use window_set_window_buttons_offset to adjust minimize/maximize/close buttons offset.

Use window_get_safe_title_margins to determine area under the title bar that is not covered by decorations.

Note: This flag is implemented only on macOS.

WindowFlags **WINDOW_FLAG_MOUSE_PASSTHROUGH** = `7`

All mouse events are passed to the underlying window of the same application.

WindowFlags **WINDOW_FLAG_MAX** = `8`

Max value of the WindowFlags.

---

enum **WindowEvent**: 🔗

WindowEvent **WINDOW_EVENT_MOUSE_ENTER** = `0`

Sent when the mouse pointer enters the window.

WindowEvent **WINDOW_EVENT_MOUSE_EXIT** = `1`

Sent when the mouse pointer exits the window.

WindowEvent **WINDOW_EVENT_FOCUS_IN** = `2`

Sent when the window grabs focus.

WindowEvent **WINDOW_EVENT_FOCUS_OUT** = `3`

Sent when the window loses focus.

WindowEvent **WINDOW_EVENT_CLOSE_REQUEST** = `4`

Sent when the user has attempted to close the window (e.g. close button is pressed).

WindowEvent **WINDOW_EVENT_GO_BACK_REQUEST** = `5`

Sent when the device "Back" button is pressed.

Note: This event is implemented only on Android.

WindowEvent **WINDOW_EVENT_DPI_CHANGE** = `6`

Sent when the window is moved to the display with different DPI, or display DPI is changed.

Note: This flag is implemented only on macOS.

WindowEvent **WINDOW_EVENT_TITLEBAR_CHANGE** = `7`

Sent when the window title bar decoration is changed (e.g. WINDOW_FLAG_EXTEND_TO_TITLE is set or window entered/exited full screen mode).

Note: This flag is implemented only on macOS.

---

enum **VSyncMode**: 🔗

VSyncMode **VSYNC_DISABLED** = `0`

No vertical synchronization, which means the engine will display frames as fast as possible (tearing may be visible). Framerate is unlimited (regardless of Engine.max_fps).

VSyncMode **VSYNC_ENABLED** = `1`

Default vertical synchronization mode, the image is displayed only on vertical blanking intervals (no tearing is visible). Framerate is limited by the monitor refresh rate (regardless of Engine.max_fps).

VSyncMode **VSYNC_ADAPTIVE** = `2`

Behaves like VSYNC_DISABLED when the framerate drops below the screen's refresh rate to reduce stuttering (tearing may be visible). Otherwise, vertical synchronization is enabled to avoid tearing. Framerate is limited by the monitor refresh rate (regardless of Engine.max_fps). Behaves like VSYNC_ENABLED when using the Compatibility rendering method.

VSyncMode **VSYNC_MAILBOX** = `3`

Displays the most recent image in the queue on vertical blanking intervals, while rendering to the other images (no tearing is visible). Framerate is unlimited (regardless of Engine.max_fps).

Although not guaranteed, the images can be rendered as fast as possible, which may reduce input lag (also called "Fast" V-Sync mode). VSYNC_MAILBOX works best when at least twice as many frames as the display refresh rate are rendered. Behaves like VSYNC_ENABLED when using the Compatibility rendering method.

---

enum **HandleType**: 🔗

HandleType **DISPLAY_HANDLE** = `0`

Display handle:

- Linux (X11): `X11::Display*` for the display.
- Android: `EGLDisplay` for the display.

HandleType **WINDOW_HANDLE** = `1`

Window handle:

- Windows: `HWND` for the window.
- Linux (X11): `X11::Window*` for the window.
- macOS: `NSWindow*` for the window.
- iOS: `UIViewController*` for the view controller.
- Android: `jObject` for the activity.

HandleType **WINDOW_VIEW** = `2`

Window view:

- Windows: `HDC` for the window (only with the GL Compatibility renderer).
- macOS: `NSView*` for the window main view.
- iOS: `UIView*` for the window main view.

HandleType **OPENGL_CONTEXT** = `3`

OpenGL context (only with the GL Compatibility renderer):

- Windows: `HGLRC` for the window (native GL), or `EGLContext` for the window (ANGLE).
- Linux (X11): `GLXContext*` for the window.
- macOS: `NSOpenGLContext*` for the window (native GL), or `EGLContext` for the window (ANGLE).
- Android: `EGLContext` for the window.

---

enum **TTSUtteranceEvent**: 🔗

TTSUtteranceEvent **TTS_UTTERANCE_STARTED** = `0`

Utterance has begun to be spoken.

TTSUtteranceEvent **TTS_UTTERANCE_ENDED** = `1`

Utterance was successfully finished.

TTSUtteranceEvent **TTS_UTTERANCE_CANCELED** = `2`

Utterance was canceled, or TTS service was unable to process it.

TTSUtteranceEvent **TTS_UTTERANCE_BOUNDARY** = `3`

Utterance reached a word or sentence boundary.

Constants

SCREEN_WITH_MOUSE_FOCUS = `-4` 🔗

Represents the screen containing the mouse pointer.

Note: On Linux (Wayland), this constant always represents the screen at index `0`.

SCREEN_WITH_KEYBOARD_FOCUS = `-3` 🔗

Represents the screen containing the window with the keyboard focus.

Note: On Linux (Wayland), this constant always represents the screen at index `0`.

SCREEN_PRIMARY = `-2` 🔗

Represents the primary screen.

Note: On Linux (Wayland), this constant always represents the screen at index `0`.

SCREEN_OF_MAIN_WINDOW = `-1` 🔗

Represents the screen where the main window is located. This is usually the default value in functions that allow specifying one of several screens.

Note: On Linux (Wayland), this constant always represents the screen at index `0`.

MAIN_WINDOW_ID = `0` 🔗

The ID of the main window spawned by the engine, which can be passed to methods expecting a `window_id`.

INVALID_WINDOW_ID = `-1` 🔗

The ID that refers to a nonexistent window. This is returned by some **DisplayServer** methods if no window matches the requested result.

INVALID_INDICATOR_ID = `-1` 🔗

The ID that refers to a nonexistent application status indicator.

Method Descriptions

String **clipboard_get**() const 🔗

Returns the user's clipboard as a string if possible.

Image **clipboard_get_image**() const 🔗

Returns the user's clipboard as an image if possible.

Note: This method uses the copied pixel data, e.g. from a image editing software or a web browser, not an image file copied from file explorer.

String **clipboard_get_primary**() const

Returns the user's primary [https://unix.stackexchange.com/questions/139191/whats-the-difference-between-primary-selection-and-clipboard-buffer] clipboard as a string if possible. This is the clipboard that is set when the user selects text in any application, rather than when pressing `Ctrl + C`. The clipboard data can then be pasted by clicking the middle mouse button in any application that supports the primary clipboard mechanism.

Note: This method is only implemented on Linux (X11/Wayland).

---

bool **clipboard_has**() const

Returns `true` if there is a text content on the user's clipboard.

---

bool **clipboard_has_image**() const

Returns `true` if there is an image content on the user's clipboard.

---

void **clipboard_set**(clipboard: String)

Sets the user's clipboard content to the given string.

---

void **clipboard_set_primary**(clipboard_primary: String)

Sets the user's primary [https://unix.stackexchange.com/questions/139191/whats-the-difference-between-primary-selection-and-clipboard-buffer] clipboard content to the given string. This is the clipboard that is set when the user selects text in any application, rather than when pressing `Ctrl + C`. The clipboard data can then be pasted by clicking the middle mouse button in any application that supports the primary clipboard mechanism.

Note: This method is only implemented on Linux (X11/Wayland).

---

int **create_status_indicator**(icon: Texture2D, tooltip: String, callback: Callable) 🔗

Creates a new application status indicator with the specified icon, tooltip, and activation callback.

`callback` should take two arguments: the pressed mouse button (one of the MouseButton constants) and the click position in screen coordinates (a Vector2i).

CursorShape **cursor_get_shape**() const 🔗

Returns the default mouse cursor shape set by cursor_set_shape.

void **cursor_set_custom_image**(cursor: Resource, shape: CursorShape = 0, hotspot: Vector2 = Vector2(0, 0)) 🔗

Sets a custom mouse cursor image for the given `shape`. This means the user's operating system and mouse cursor theme will no longer influence the mouse cursor's appearance.

`cursor` can be either a Texture2D or an Image, and it should not be larger than 256×256 to display correctly. Optionally, `hotspot` can be set to offset the image's position relative to the click point. By default, `hotspot` is set to the top-left corner of the image. See also cursor_set_shape.

void **cursor_set_shape**(shape: CursorShape) 🔗

Sets the default mouse cursor shape. The cursor's appearance will vary depending on the user's operating system and mouse cursor theme. See also cursor_get_shape and cursor_set_custom_image.

void **delete_status_indicator**(id: int) 🔗

Removes the application status indicator.

Error **dialog_input_text**(title: String, description: String, existing_text: String, callback: Callable) 🔗

Shows a text input dialog which uses the operating system's native look-and-feel. `callback` should accept a single String parameter which contains the text field's contents.

Note: This method is implemented if the display server has the FEATURE_NATIVE_DIALOG_INPUT feature. Supported platforms include macOS and Windows.

Error **dialog_show**(title: String, description: String, buttons: PackedStringArray, callback: Callable) 🔗

Shows a text dialog which uses the operating system's native look-and-feel. `callback` should accept a single int parameter which corresponds to the index of the pressed button.

Note: This method is implemented if the display server has the FEATURE_NATIVE_DIALOG feature. Supported platforms include macOS and Windows.

---

void **enable_for_stealing_focus**(process_id: int) 🔗

Allows the `process_id` PID to steal focus from this window. In other words, this disables the operating system's focus stealing protection for the specified PID.

Note: This method is implemented only on Windows.

Error **file_dialog_show**(title: String, current_directory: String, filename: String, show_hidden: bool, mode: FileDialogMode, filters: PackedStringArray, callback: Callable) 🔗

Displays OS native dialog for selecting files or directories in the file system.

Each filter string in the `filters` array should be formatted like this: `*.txt,*.doc;Text Files`. The description text of the filter is optional and can be omitted. See also FileDialog.filters.

Callbacks have the following arguments: `status: bool, selected_paths: PackedStringArray, selected_filter_index: int`.

Note: This method is implemented if the display server has the FEATURE_NATIVE_DIALOG_FILE feature. Supported platforms include Linux

(X11/Wayland), Windows, and macOS.

Note: `current_directory` might be ignored.

Note: On Linux, `show_hidden` is ignored.

Note: On macOS, native file dialogs have no title.

Note: On macOS, sandboxed apps will save security-scoped bookmarks to retain access to the opened folders across multiple sessions. Use OS.get_granted_permissions to get a list of saved bookmarks.

Error **file_dialog_with_options_show**(title: String, current_directory: String, root: String, filename: String, show_hidden: bool, mode: FileDialogMode, filters: PackedStringArray, options: Array[Dictionary], callback: Callable) 🔗

Displays OS native dialog for selecting files or directories in the file system with additional user selectable options.

Each filter string in the `filters` array should be formatted like this: `*.txt,*.doc;Text Files`. The description text of the filter is optional and can be omitted. See also FileDialog.filters.

`options` is array of Dictionarys with the following keys:

- `"name"` - option's name String.
- `"values"` - PackedStringArray of values. If empty, boolean option (check box) is used.
- `"default"` - default selected option index (int) or default boolean value (bool).

Callbacks have the following arguments: `status: bool, selected_paths: PackedStringArray, selected_filter_index: int, selected_option: Dictionary`.

Note: This method is implemented if the display server has the FEATURE_NATIVE_DIALOG_FILE feature. Supported platforms include Linux (X11/Wayland), Windows, and macOS.

Note: `current_directory` might be ignored.

Note: On Linux (X11), `show_hidden` is ignored.

Note: On macOS, native file dialogs have no title.

Note: On macOS, sandboxed apps will save security-scoped bookmarks to retain access to the opened folders across multiple sessions. Use OS.get_granted_permissions to get a list of saved bookmarks.

---

void **force_process_and_drop_events**()

Forces window manager processing while ignoring all InputEvents. See also process_events.

Note: This method is implemented on Windows and macOS.

Color **get_accent_color**() const

Returns OS theme accent color. Returns `Color(0, 0, 0, 0)`, if accent color is unknown.

Note: This method is implemented on macOS and Windows.

Color **get_base_color**() const

Returns the OS theme base color (default control background). Returns `Color(0, 0, 0, 0)` if the base color is unknown.

Note: This method is implemented on macOS and Windows.

Array[Rect2] **get_display_cutouts**() const

Returns an Array of Rect2, each of which is the bounding rectangle for a display cutout or notch. These are non-functional areas on edge-to-edge screens used by cameras and sensors. Returns an empty array if the device does not have cutouts. See also get_display_safe_area.

Note: Currently only implemented on Android. Other platforms will return an empty array even if they do have display cutouts or notches.

Rect2i **get_display_safe_area**() const

Returns the unobscured area of the display where interactive controls should be rendered. See also get_display_cutouts.

---

int **get_keyboard_focus_screen**() const

Returns the index of the screen containing the window with the keyboard focus, or the primary screen if there's no focused window.

String **get_name**() const

Returns the name of the **DisplayServer** currently in use. Most operating systems only have a single **DisplayServer**, but Linux has access to more than one **DisplayServer** (currently X11 and Wayland).

The names of built-in display servers are `Windows`, `macOS`, `X11` (Linux), `Wayland` (Linux), `Android`, `iOS`, `web` (HTML5), and `headless` (when started with the `--headless` command line argument).

---

int **get_primary_screen**() const

Returns index of the primary screen.

---

int **get_screen_count**() const

Returns the number of displays available.

---

int **get_screen_from_rect**(rect: Rect2) const

Returns index of the screen which contains specified rectangle.

---

bool **get_swap_cancel_ok**()

Returns `true` if positions of **OK** and **Cancel** buttons are swapped in dialogs. This is enabled by default on Windows to follow interface conventions, and be toggled by changing ProjectSettings.gui/common/swap_cancel_ok.

Note: This doesn't affect native dialogs such as the ones spawned by dialog_show.

---

int **get_window_at_screen_position**(position: Vector2i) const

Returns the ID of the window at the specified screen `position` (in pixels). On multi-monitor setups, the screen position is relative to the virtual desktop area. On multi-monitor setups with different screen resolutions or orientations, the origin may be located outside any display like this:

```
* (0, 0)        +-------+
                |       |
+-------------+ |       |
|             | |       |
|             | |       |
+-------------+ +-------+
```

PackedInt32Array **get_window_list**() const 🔗

Returns the list of Godot window IDs belonging to this process.

Note: Native dialogs are not included in this list.

int **global_menu_add_check_item**(menu_root: String, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Adds a new checkable item with text `label` to the global menu with ID `menu_root`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
```

```
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

---

int **global_menu_add_icon_check_item**(menu_root: String, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)

Deprecated: Use NativeMenu or PopupMenu instead.

Adds a new checkable item with text `label` and icon `icon` to the global menu with ID `menu_root`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

---

int **global_menu_add_icon_item**(menu_root: String, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)

Deprecated: Use NativeMenu or PopupMenu instead.

Adds a new item with text `label` and icon `icon` to the global menu with ID `menu_root`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

---

int **global_menu_add_icon_radio_check_item**(menu_root: String, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Adds a new radio-checkable item with text `label` and icon `icon` to the global menu with ID `menu_root`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: Radio-checkable items just display a checkmark, but don't have any built-in checking behavior and must be checked/unchecked manually. See global_menu_set_item_checked for more info on how to control it.

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

---

int **global_menu_add_item**(menu_root: String, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)

Deprecated: Use NativeMenu or PopupMenu instead.

Adds a new item with text `label` to the global menu with ID `menu_root`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

---

int **global_menu_add_multistate_item**(menu_root: String, label: String, max_states: int, default_state: int, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)

Deprecated: Use NativeMenu or PopupMenu instead.

Adds a new item with text `label` to the global menu with ID `menu_root`.

Contrarily to normal binary items, multistate items can have more than two states, as defined by `max_states`. Each press or activate of the item will increase the state by one. The default value is defined by `default_state`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: By default, there's no indication of the current item state, it should be changed manually.

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

---

int **global_menu_add_radio_check_item**(menu_root: String, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)

Deprecated: Use NativeMenu or PopupMenu instead.

Adds a new radio-checkable item with text `label` to the global menu with ID `menu_root`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: Radio-checkable items just display a checkmark, but don't have any built-in checking behavior and must be checked/unchecked manually. See global_menu_set_item_checked for more info on how to control it.

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

---

int **global_menu_add_separator**(menu_root: String, index: int = -1)

Deprecated: Use NativeMenu or PopupMenu instead.

Adds a separator between items to the global menu with ID `menu_root`. Separators also occupy an index.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

---

int **global_menu_add_submenu_item**(menu_root: String, label: String, submenu: String, index: int = -1) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Adds an item that will act as a submenu of the global menu `menu_root`. The `submenu` argument is the ID of the global menu root that will be shown when the item is clicked.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

---

void **global_menu_clear**(menu_root: String) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Removes all items from the global menu with ID `menu_root`.

Note: This method is implemented only on macOS.

Supported system menu IDs:

```
"_main" - Main menu (macOS).
"_dock" - Dock popup menu (macOS).
"_apple" - Apple menu (macOS, custom items added before "Services").
"_window" - Window menu (macOS, custom items added after "Bring All to
Front").
"_help" - Help menu (macOS).
```

Key **global_menu_get_item_accelerator**(menu_root: String, idx: int) const 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the accelerator of the item at index `idx`. Accelerators are special combinations of keys that activate the item, no matter which control is focused.

Note: This method is implemented only on macOS.

Callable **global_menu_get_item_callback**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the callback of the item at index `idx`.

Note: This method is implemented only on macOS.

---

int **global_menu_get_item_count**(menu_root: String) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns number of items in the global menu with ID `menu_root`.

Note: This method is implemented only on macOS.

Texture2D **global_menu_get_item_icon**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the icon of the item at index `idx`.

Note: This method is implemented only on macOS.

---

int **global_menu_get_item_indentation_level**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the horizontal offset of the item at the given `idx`.

Note: This method is implemented only on macOS.

---

int **global_menu_get_item_index_from_tag**(menu_root: String, tag: Variant) const 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the index of the item with the specified `tag`. Indices are automatically assigned to each item by the engine, and cannot be set manually.

Note: This method is implemented only on macOS.

---

int **global_menu_get_item_index_from_text**(menu_root: String, text: String) const 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the index of the item with the specified `text`. Indices are automatically assigned to each item by the engine, and cannot be set manually.

Note: This method is implemented only on macOS.

Callable **global_menu_get_item_key_callback**(menu_root: String, idx: int) const 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the callback of the item accelerator at index `idx`.

Note: This method is implemented only on macOS.

---

int **global_menu_get_item_max_states**(menu_root: String, idx: int) const 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Returns number of states of a multistate item. See global_menu_add_multistate_item for details.

Note: This method is implemented only on macOS.

---

int **global_menu_get_item_state**(menu_root: String, idx: int) const 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the state of a multistate item. See global_menu_add_multistate_item for details.

Note: This method is implemented only on macOS.

String **global_menu_get_item_submenu**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the submenu ID of the item at index `idx`. See global_menu_add_submenu_item for more info on how to add a submenu.

Note: This method is implemented only on macOS.

Variant **global_menu_get_item_tag**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the metadata of the specified item, which might be of any type. You can set it with global_menu_set_item_tag, which provides a simple way of assigning context data to items.

Note: This method is implemented only on macOS.

String **global_menu_get_item_text**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the text of the item at index `idx`.

Note: This method is implemented only on macOS.

String **global_menu_get_item_tooltip**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns the tooltip associated with the specified index `idx`.

Note: This method is implemented only on macOS.

Dictionary **global_menu_get_system_menu_roots**() const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns Dictionary of supported system menu IDs and names.

Note: This method is implemented only on macOS.

bool **global_menu_is_item_checkable**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns `true` if the item at index `idx` is checkable in some way, i.e. if it has a checkbox or radio button.

Note: This method is implemented only on macOS.

bool **global_menu_is_item_checked**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns `true` if the item at index `idx` is checked.

Note: This method is implemented only on macOS.

bool **global_menu_is_item_disabled**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns `true` if the item at index `idx` is disabled. When it is disabled it can't be selected, or its action invoked.

See global_menu_set_item_disabled for more info on how to disable an item.

Note: This method is implemented only on macOS.

bool **global_menu_is_item_hidden**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns `true` if the item at index `idx` is hidden.

See global_menu_set_item_hidden for more info on how to hide an item.

Note: This method is implemented only on macOS.

---

bool **global_menu_is_item_radio_checkable**(menu_root: String, idx: int) const

Deprecated: Use NativeMenu or PopupMenu instead.

Returns `true` if the item at index `idx` has radio button-style checkability.

Note: This is purely cosmetic; you must add the logic for checking/unchecking items in radio groups.

Note: This method is implemented only on macOS.

---

void **global_menu_remove_item**(menu_root: String, idx: int)

Deprecated: Use NativeMenu or PopupMenu instead.

Removes the item at index `idx` from the global menu `menu_root`.

Note: The indices of items after the removed item will be shifted by one.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_accelerator**(menu_root: String, idx: int, keycode: Key)

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the accelerator of the item at index `idx`. `keycode` can be a single Key, or a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_callback**(menu_root: String, idx: int, callback: Callable)

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the callback of the item at index `idx`. Callback is emitted when an item is pressed.

Note: The `callback` Callable needs to accept exactly one Variant parameter, the parameter passed to the Callable will be the value passed to the `tag` parameter when the menu item was created.

Note: This method is implemented only on macOS.

void **global_menu_set_item_checkable**(menu_root: String, idx: int, checkable: bool)

Deprecated: Use NativeMenu or PopupMenu instead.

Sets whether the item at index `idx` has a checkbox. If `false`, sets the type of the item to plain text.

Note: This method is implemented only on macOS.

void **global_menu_set_item_checked**(menu_root: String, idx: int, checked: bool)

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the checkstate status of the item at index `idx`.

Note: This method is implemented only on macOS.

void **global_menu_set_item_disabled**(menu_root: String, idx: int, disabled: bool)

Deprecated: Use NativeMenu or PopupMenu instead.

Enables/disables the item at index `idx`. When it is disabled, it can't be selected and its action can't be invoked.

Note: This method is implemented only on macOS.

void **global_menu_set_item_hidden**(menu_root: String, idx: int, hidden: bool)

Deprecated: Use NativeMenu or PopupMenu instead.

Hides/shows the item at index `idx`. When it is hidden, an item does not appear in a menu and its action cannot be invoked.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_hover_callbacks**(menu_root: String, idx: int, callback: Callable) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the callback of the item at index `idx`. The callback is emitted when an item is hovered.

Note: The `callback` Callable needs to accept exactly one Variant parameter, the parameter passed to the Callable will be the value passed to the `tag` parameter when the menu item was created.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_icon**(menu_root: String, idx: int, icon: Texture2D) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Replaces the Texture2D icon of the specified `idx`.

Note: This method is implemented only on macOS.

Note: This method is not supported by macOS "_dock" menu items.

---

void **global_menu_set_item_indentation_level**(menu_root: String, idx: int, level: int) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the horizontal offset of the item at the given `idx`.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_key_callback**(menu_root: String, idx: int, key_callback: Callable)

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the callback of the item at index `idx`. Callback is emitted when its accelerator is activated.

Note: The `key_callback` Callable needs to accept exactly one Variant parameter, the parameter passed to the Callable will be the value passed to the `tag` parameter when the menu item was created.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_max_states**(menu_root: String, idx: int, max_states: int)

Deprecated: Use NativeMenu or PopupMenu instead.

Sets number of state of a multistate item. See global_menu_add_multistate_item for details.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_radio_checkable**(menu_root: String, idx: int, checkable: bool)

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the type of the item at the specified index `idx` to radio button. If `false`, sets the type of the item to plain text.

Note: This is purely cosmetic; you must add the logic for checking/unchecking items in radio groups.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_state**(menu_root: String, idx: int, state: int)

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the state of a multistate item. See global_menu_add_multistate_item for details.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_submenu**(menu_root: String, idx: int, submenu: String) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the submenu of the item at index `idx`. The submenu is the ID of a global menu root that would be shown when the item is clicked.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_tag**(menu_root: String, idx: int, tag: Variant) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the metadata of an item, which may be of any type. You can later get it with global_menu_get_item_tag, which provides a simple way of assigning context data to items.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_text**(menu_root: String, idx: int, text: String) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the text of the item at index `idx`.

Note: This method is implemented only on macOS.

---

void **global_menu_set_item_tooltip**(menu_root: String, idx: int, tooltip: String) 🔗

Deprecated: Use NativeMenu or PopupMenu instead.

Sets the String tooltip of the item at the specified index `idx`.

Note: This method is implemented only on macOS.

---

void **global_menu_set_popup_callbacks**(menu_root: String, open_callback: Callable, close_callback: Callable)

Deprecated: Use NativeMenu or PopupMenu instead.

Registers callables to emit when the menu is respectively about to show or closed. Callback methods should have zero arguments.

---

bool **has_additional_outputs**() const

Returns `true` if any additional outputs have been registered via register_additional_output.

---

bool **has_feature**(feature: Feature) const

Returns `true` if the specified `feature` is supported by the current **DisplayServer**, `false` otherwise.

---

void **help_set_search_callbacks**(search_callback: Callable, action_callback: Callable)

Sets native help system search callbacks.

`search_callback` has the following arguments: `String search_string, int result_limit` and return a Dictionary with "key, display name" pairs for the search results. Called when the user enters search terms in the `Help` menu.

`action_callback` has the following arguments: `String key`. Called when the user selects a search result in the `Help` menu.

Note: This method is implemented only on macOS.

Vector2i **ime_get_selection**() const

Returns the text selection in the Input Method Editor [https://en.wikipedia.org/wiki/Input_method] composition string, with the Vector2i's `x` component being the caret position and `y` being the length of the selection.

Note: This method is implemented only on macOS.

String **ime_get_text**() const

Returns the composition string contained within the Input Method Editor [https://en.wikipedia.org/wiki/Input_method] window.

Note: This method is implemented only on macOS.

---

bool **is_dark_mode**() const

Returns `true` if OS is using dark mode.

Note: This method is implemented on Android, iOS, macOS, Windows, and Linux (X11/Wayland).

---

bool **is_dark_mode_supported**() const

Returns `true` if OS supports dark mode.

Note: This method is implemented on Android, iOS, macOS, Windows, and Linux (X11/Wayland).

---

bool **is_touchscreen_available**() const

Returns `true` if touch events are available (Android or iOS), the capability is detected on the Web platform or if ProjectSettings.input_devices/pointing/emulate_touch_from_mouse is `true`.

---

bool **is_window_transparency_available**() const

Returns `true` if the window background can be made transparent. This method returns `false` if ProjectSettings.display/window/per_pixel_transparency/allowed is set to `false`, or if transparency is not supported by the renderer or OS compositor.

---

int **keyboard_get_current_layout**() const

Returns active keyboard layout index.

Note: This method is implemented on Linux (X11/Wayland), macOS, and Windows.

Key **keyboard_get_keycode_from_physical**(keycode: Key) const

Converts a physical (US QWERTY) `keycode` to one in the active keyboard layout.

Note: This method is implemented on Linux (X11/Wayland), macOS and Windows.

Key **keyboard_get_label_from_physical**(keycode: Key) const

Converts a physical (US QWERTY) `keycode` to localized label printed on the key in the active keyboard layout.

Note: This method is implemented on Linux (X11/Wayland), macOS and Windows.

---

int **keyboard_get_layout_count**() const

Returns the number of keyboard layouts.

Note: This method is implemented on Linux (X11/Wayland), macOS and Windows.

String **keyboard_get_layout_language**(index: int) const

Returns the ISO-639/BCP-47 language code of the keyboard layout at position `index`.

Note: This method is implemented on Linux (X11/Wayland), macOS and Windows.

String **keyboard_get_layout_name**(index: int) const

Returns the localized name of the keyboard layout at position `index`.

Note: This method is implemented on Linux (X11/Wayland), macOS and Windows.

---

void **keyboard_set_current_layout**(index: int)

Sets the active keyboard layout.

Note: This method is implemented on Linux (X11/Wayland), macOS and Windows.

BitField[MouseButtonMask] **mouse_get_button_state**() const

Returns the current state of mouse buttons (whether each button is pressed) as a bitmask. If multiple mouse buttons are pressed at the same time, the bits are added together. Equivalent to Input.get_mouse_button_mask.

MouseMode **mouse_get_mode**() const

Returns the current mouse mode. See also mouse_set_mode.

Vector2i **mouse_get_position**() const

Returns the mouse cursor's current position in screen coordinates.

---

void **mouse_set_mode**(mouse_mode: MouseMode)

Sets the current mouse mode. See also mouse_get_mode.

---

void **process_events**()

Perform window manager processing, including input flushing. See also force_process_and_drop_events, Input.flush_buffered_events and Input.use_accumulated_input.

---

void **register_additional_output**(object: Object)

Registers an Object which represents an additional output that will be rendered too, beyond normal windows. The Object is only used as an identifier, which can be later passed to unregister_additional_output.

This can be used to prevent Godot from skipping rendering when no normal windows are visible.

---

int **screen_get_dpi**(screen: int = -1) const

Returns the dots per inch density of the specified screen. If `screen` is SCREEN_OF_MAIN_WINDOW (the default value), a screen with the main window will be used.

Note: On macOS, returned value is inaccurate if fractional display scaling mode is used.

Note: On Android devices, the actual screen densities are grouped into six generalized densities:

```
   ldpi - 120 dpi
   mdpi - 160 dpi
   hdpi - 240 dpi
  xhdpi - 320 dpi
 xxhdpi - 480 dpi
xxxhdpi - 640 dpi
```

Note: This method is implemented on Android, Linux (X11/Wayland), macOS and Windows. Returns `72` on unsupported platforms.

Image **screen_get_image**(screen: int = -1) const

Returns screenshot of the `screen`.

Note: This method is implemented on Linux (X11), macOS, and Windows.

Note: On macOS, this method requires "Screen Recording" permission, if permission is not granted it will return desktop wallpaper color.

float **screen_get_max_scale**() const

Returns the greatest scale factor of all screens.

Note: On macOS returned value is `2.0` if there is at least one hiDPI (Retina) screen in the system, and `1.0` in all other cases.

Note: This method is implemented only on macOS.

ScreenOrientation **screen_get_orientation**(screen: int = -1) const

Returns the `screen`'s current orientation. See also screen_set_orientation.

Note: This method is implemented on Android and iOS.

Color **screen_get_pixel**(position: Vector2i) const

Returns color of the display pixel at the `position`.

Note: This method is implemented on Linux (X11), macOS, and Windows.

Note: On macOS, this method requires "Screen Recording" permission, if permission is not granted it will return desktop wallpaper color.

Vector2i **screen_get_position**(screen: int = -1) const

Returns the screen's top-left corner position in pixels. On multi-monitor setups, the screen position is relative to the virtual desktop area. On multi-monitor setups with different screen resolutions or orientations, the origin may be located outside any display like this:

```
* (0, 0)         +--------+
                 |        |
+--------------+ |        |
|              | |        |
|              | |        |
+--------------+ +--------+
```

See also screen_get_size.

Note: On Linux (Wayland) this method always returns `(0, 0)`.

float **screen_get_refresh_rate**(screen: int = -1) const

Returns the current refresh rate of the specified screen. If `screen` is SCREEN_OF_MAIN_WINDOW (the default value), a screen with the main window will be used.

Note: Returns `-1.0` if the DisplayServer fails to find the refresh rate for the specified screen. On Web, screen_get_refresh_rate will always return `-1.0` as there is no way to retrieve the refresh rate on that platform.

To fallback to a default refresh rate if the method fails, try:

```
var refresh_rate = DisplayServer.screen_get_refresh_rate()
if refresh_rate < 0:
    refresh_rate = 60.0
```

float **screen_get_scale**(screen: int = -1) const

Returns the scale factor of the specified screen by index.

Note: On macOS, the returned value is `2.0` for hiDPI (Retina) screens, and `1.0` for all other cases.

Note: On Linux (Wayland), the returned value is accurate only when `screen` is SCREEN_OF_MAIN_WINDOW. Due to API limitations, passing a direct index will return a rounded-up integer, if the screen has a fractional scale (e.g. `1.25` would get rounded up to `2.0`).

Note: This method is implemented only on macOS and Linux (Wayland).

Vector2i **screen_get_size**(screen: int = -1) const

Returns the screen's size in pixels. See also screen_get_position and screen_get_usable_rect.

Rect2i **screen_get_usable_rect**(screen: int = -1) const

Returns the portion of the screen that is not obstructed by a status bar in pixels. See also screen_get_size.

bool **screen_is_kept_on**() const

Returns `true` if the screen should never be turned off by the operating system's power-saving measures. See also screen_set_keep_on.

void **screen_set_keep_on**(enable: bool)

Sets whether the screen should never be turned off by the operating system's power-saving measures. See also screen_is_kept_on.

void **screen_set_orientation**(orientation: ScreenOrientation, screen: int = -1)

Sets the `screen`'s `orientation`. See also screen_get_orientation.

Note: On iOS, this method has no effect if ProjectSettings.display/window/handheld/orientation is not set to SCREEN_SENSOR.

---

void **set_icon**(image: Image) 🔗

Sets the window icon (usually displayed in the top-left corner) with an Image. To use icons in the operating system's native format, use set_native_icon instead.

Note: Requires support for FEATURE_ICON.

---

void **set_native_icon**(filename: String) 🔗

Sets the window icon (usually displayed in the top-left corner) in the operating system's *native* format. The file at `filename` must be in `.ico` format on Windows or `.icns` on macOS. By using specially crafted `.ico` or `.icns` icons, set_native_icon allows specifying different icons depending on the size the icon is displayed at. This size is determined by the operating system and user preferences (including the display scale factor). To use icons in other formats, use set_icon instead.

Note: Requires support for FEATURE_NATIVE_ICON.

---

void **set_system_theme_change_callback**(callable: Callable) 🔗

Sets the `callable` that should be called when system theme settings are changed. Callback method should have zero arguments.

Note: This method is implemented on Android, iOS, macOS, Windows, and Linux (X11/Wayland).

Rect2 **status_indicator_get_rect**(id: int) const 🔗

Returns the rectangle for the given status indicator `id` in screen coordinates. If the status indicator is not visible, returns an empty Rect2.

Note: This method is implemented on macOS and Windows.

---

void **status_indicator_set_callback**(id: int, callback: Callable) 🔗

Sets the application status indicator activation callback. `callback` should take two arguments: int mouse button index (one of MouseButton values) and Vector2i click position in screen coordinates.

Note: This method is implemented on macOS and Windows.

---

void **status_indicator_set_icon**(id: int, icon: Texture2D) 🔗

Sets the application status indicator icon.

Note: This method is implemented on macOS and Windows.

---

void **status_indicator_set_menu**(id: int, menu_rid: RID) 🔗

Sets the application status indicator native popup menu.

Note: On macOS, the menu is activated by any mouse button. Its activation callback is *not* triggered.

Note: On Windows, the menu is activated by the right mouse button, selecting the status icon and pressing `Shift + F10`, or the applications key. The menu's activation callback for the other mouse buttons is still triggered.

Note: Native popup is only supported if NativeMenu supports the NativeMenu.FEATURE_POPUP_MENU feature.

---

void **status_indicator_set_tooltip**(id: int, tooltip: String) 🔗

Sets the application status indicator tooltip.

Note: This method is implemented on macOS and Windows.

String **tablet_get_current_driver**() const 🔗

Returns current active tablet driver name.

Note: This method is implemented only on Windows.

---

int **tablet_get_driver_count**() const 🔗

Returns the total number of available tablet drivers.

Note: This method is implemented only on Windows.

String **tablet_get_driver_name**(idx: int) const

Returns the tablet driver name for the given index.

Note: This method is implemented only on Windows.

---

void **tablet_set_current_driver**(name: String)

Set active tablet driver name.

Supported drivers:

- `winink`: Windows Ink API, default (Windows 8.1+ required).
- `wintab`: Wacom Wintab API (compatible device driver required).
- `dummy`: Dummy driver, tablet input is disabled.

Note: This method is implemented only on Windows.

Array[Dictionary] **tts_get_voices**() const

Returns an Array of voice information dictionaries.

Each Dictionary contains two String entries:

- `name` is voice name.
- `id` is voice identifier.
- `language` is language code in `lang_Variant` format. The `lang` part is a 2 or 3-letter code based on the ISO-639 standard, in lowercase. The `Variant` part is an engine-dependent string describing country, region or/and dialect.

Note that Godot depends on system libraries for text-to-speech functionality. These libraries are installed by default on Windows and macOS, but not on all Linux distributions. If they are not present, this method will return an empty list. This applies to both Godot users on Linux, as well as end-users on Linux running Godot games that use text-to-speech.

Note: This method is implemented on Android, iOS, Web, Linux (X11/Wayland), macOS, and Windows.

Note: ProjectSettings.audio/general/text_to_speech should be `true` to use text-to-speech.

PackedStringArray **tts_get_voices_for_language**(language: String) const

Returns an PackedStringArray of voice identifiers for the `language`.

Note: This method is implemented on Android, iOS, Web, Linux (X11/Wayland), macOS, and Windows.

Note: ProjectSettings.audio/general/text_to_speech should be `true` to use text-to-speech.

---

bool **tts_is_paused**() const

Returns `true` if the synthesizer is in a paused state.

Note: This method is implemented on Android, iOS, Web, Linux (X11/Wayland), macOS, and Windows.

Note: ProjectSettings.audio/general/text_to_speech should be `true` to use text-to-speech.

---

bool **tts_is_speaking**() const

Returns `true` if the synthesizer is generating speech, or have utterance waiting in the queue.

Note: This method is implemented on Android, iOS, Web, Linux (X11/Wayland), macOS, and Windows.

Note: ProjectSettings.audio/general/text_to_speech should be `true` to use text-to-speech.

---

void **tts_pause**()

Puts the synthesizer into a paused state.

Note: This method is implemented on Android, iOS, Web, Linux (X11/Wayland), macOS, and Windows.

Note: ProjectSettings.audio/general/text_to_speech should be `true` to use text-to-speech.

---

void **tts_resume**()

Resumes the synthesizer if it was paused.

Note: This method is implemented on Android, iOS, Web, Linux (X11/Wayland), macOS, and Windows.

Note: ProjectSettings.audio/general/text_to_speech should be `true` to use text-to-speech.

---

void **tts_set_utterance_callback**(event: TTSUtteranceEvent, callable: Callable)

Adds a callback, which is called when the utterance has started, finished, canceled or reached a text boundary.

- TTS_UTTERANCE_STARTED, TTS_UTTERANCE_ENDED, and TTS_UTTERANCE_CANCELED callable's method should take one int parameter, the utterance ID.
- TTS_UTTERANCE_BOUNDARY callable's method should take two int parameters, the index of the character and the utterance ID.

Note: The granularity of the boundary callbacks is engine dependent.

Note: This method is implemented on Android, iOS, Web, Linux (X11/Wayland), macOS, and Windows.

Note: ProjectSettings.audio/general/text_to_speech should be `true` to use text-to-speech.

---

void **tts_speak**(text: String, voice: String, volume: int = 50, pitch: float = 1.0, rate: float = 1.0, utterance_id: int = 0, interrupt: bool = false)

Adds an utterance to the queue. If `interrupt` is `true`, the queue is cleared first.

- `voice` identifier is one of the `"id"` values returned by tts_get_voices or one of the values returned by tts_get_voices_for_language.
- `volume` ranges from `0` (lowest) to `100` (highest).

- `pitch` ranges from `0.0` (lowest) to `2.0` (highest), `1.0` is default pitch for the current voice.
- `rate` ranges from `0.1` (lowest) to `10.0` (highest), `1.0` is a normal speaking rate. Other values act as a percentage relative.
- `utterance_id` is passed as a parameter to the callback functions.

Note: On Windows and Linux (X11/Wayland), utterance `text` can use SSML markup. SSML support is engine and voice dependent. If the engine does not support SSML, you should strip out all XML markup before calling tts_speak.

Note: The granularity of pitch, rate, and volume is engine and voice dependent. Values may be truncated.

Note: This method is implemented on Android, iOS, Web, Linux (X11/Wayland), macOS, and Windows.

Note: ProjectSettings.audio/general/text_to_speech should be `true` to use text-to-speech.

---

void **tts_stop**()

Stops synthesis in progress and removes all utterances from the queue.

Note: This method is implemented on Android, iOS, Web, Linux (X11/Linux), macOS, and Windows.

Note: ProjectSettings.audio/general/text_to_speech should be `true` to use text-to-speech.

---

void **unregister_additional_output**(object: Object)

Unregisters an Object representing an additional output, that was registered via register_additional_output.

---

int **virtual_keyboard_get_height**() const

Returns the on-screen keyboard's height in pixels. Returns 0 if there is no keyboard or if it is currently hidden.

---

void **virtual_keyboard_hide**()

Hides the virtual keyboard if it is shown, does nothing otherwise.

---

void **virtual_keyboard_show**(existing_text: String, position: Rect2 = Rect2(0, 0, 0, 0), type: VirtualKeyboardType = 0, max_length: int = -1, cursor_start: int = -1, cursor_end: int = -1)

Shows the virtual keyboard if the platform has one.

`existing_text` parameter is useful for implementing your own LineEdit or TextEdit, as it tells the virtual keyboard what text has already been typed (the virtual keyboard uses it for auto-correct and predictions).

`position` parameter is the screen space Rect2 of the edited text.

`type` parameter allows configuring which type of virtual keyboard to show.

`max_length` limits the number of characters that can be entered if different from `-1`.

`cursor_start` can optionally define the current text cursor position if `cursor_end` is not set.

`cursor_start` and `cursor_end` can optionally define the current text selection.

Note: This method is implemented on Android, iOS and Web.

---

void **warp_mouse**(position: Vector2i)

Sets the mouse cursor position to the given `position` relative to an origin at the upper left corner of the currently focused game Window Manager window.

Note: warp_mouse is only supported on Windows, macOS, and Linux (X11/Wayland). It has no effect on Android, iOS, and Web.

---

bool **window_can_draw**(window_id: int = 0) const

Returns `true` if anything can be drawn in the window specified by `window_id`, `false` otherwise. Using the `--disable-render-loop` command line argument or a headless build will return `false`.

---

int **window_get_active_popup**() const

Returns ID of the active popup window, or INVALID_WINDOW_ID if there is none.

---

int **window_get_attached_instance_id**(window_id: int = 0) const

Returns the Object.get_instance_id of the Window the `window_id` is attached to.

---

int **window_get_current_screen**(window_id: int = 0) const

Returns the screen the window specified by `window_id` is currently positioned on. If the screen overlaps multiple displays, the screen where the window's center is located is returned. See also window_set_current_screen.

---

bool **window_get_flag**(flag: WindowFlags, window_id: int = 0) const

Returns the current value of the given window's `flag`.

Vector2i **window_get_max_size**(window_id: int = 0) const

Returns the window's maximum size (in pixels). See also window_set_max_size.

Vector2i **window_get_min_size**(window_id: int = 0) const

Returns the window's minimum size (in pixels). See also window_set_min_size.

WindowMode **window_get_mode**(window_id: int = 0) const

Returns the mode of the given window.

---

int **window_get_native_handle**(handle_type: HandleType, window_id: int = 0) const

Returns internal structure pointers for use in plugins.

Note: This method is implemented on Android, Linux (X11/Wayland), macOS, and Windows.

Rect2i **window_get_popup_safe_rect**(window: int) const

Returns the bounding box of control, or menu item that was used to open the popup window, in the screen coordinate system.

Vector2i **window_get_position**(window_id: int = 0) const

Returns the position of the client area of the given window on the screen.

Vector2i **window_get_position_with_decorations**(window_id: int = 0) const

Returns the position of the given window on the screen including the borders drawn by the operating system. See also window_get_position.

Vector3i **window_get_safe_title_margins**(window_id: int = 0) const

Returns left margins (`x`), right margins (`y`) and height (`z`) of the title that are safe to use (contains no buttons or other elements) when WINDOW_FLAG_EXTEND_TO_TITLE flag is set.

Vector2i **window_get_size**(window_id: int = 0) const

Returns the size of the window specified by `window_id` (in pixels), excluding the borders drawn by the operating system. This is also called the "client area". See also window_get_size_with_decorations, window_set_size and window_get_position.

Vector2i **window_get_size_with_decorations**(window_id: int = 0) const

Returns the size of the window specified by `window_id` (in pixels), including the borders drawn by the operating system. See also window_get_size.

Vector2i **window_get_title_size**(title: String, window_id: int = 0) const

Returns the estimated window title bar size (including text and window buttons) for the window specified by `window_id` (in pixels). This method does not change the window title.

Note: This method is implemented on macOS and Windows.

VSyncMode **window_get_vsync_mode**(window_id: int = 0) const

Returns the V-Sync mode of the given window.

bool **window_is_focused**(window_id: int = 0) const

Returns `true` if the window specified by `window_id` is focused.

bool **window_is_maximize_allowed**(window_id: int = 0) const

Returns `true` if the given window can be maximized (the maximize button is enabled).

bool **window_maximize_on_title_dbl_click**() const

Returns `true`, if double-click on a window title should maximize it.

Note: This method is implemented only on macOS.

bool **window_minimize_on_title_dbl_click**() const

Returns `true`, if double-click on a window title should minimize it.

Note: This method is implemented only on macOS.

void **window_move_to_foreground**(window_id: int = 0)

Moves the window specified by `window_id` to the foreground, so that it is visible over other windows.

void **window_request_attention**(window_id: int = 0)

Makes the window specified by `window_id` request attention, which is materialized by the window title and taskbar entry blinking until the window is focused. This usually has no visible effect if the window is currently focused. The exact behavior varies depending on the operating system.

---

void **window_set_current_screen**(screen: int, window_id: int = 0)

Moves the window specified by `window_id` to the specified `screen`. See also window_get_current_screen.

---

void **window_set_drop_files_callback**(callback: Callable, window_id: int = 0)

Sets the `callback` that should be called when files are dropped from the operating system's file manager to the window specified by `window_id`. `callback` should take one PackedStringArray argument, which is the list of dropped files.

Warning: Advanced users only! Adding such a callback to a Window node will override its default implementation, which can introduce bugs.

Note: This method is implemented on Windows, macOS, Linux (X11/Wayland), and Web.

---

void **window_set_exclusive**(window_id: int, exclusive: bool)

If set to `true`, this window will always stay on top of its parent window, parent window will ignore input while this window is opened.

Note: On macOS, exclusive windows are confined to the same space (virtual desktop or screen) as the parent window.

Note: This method is implemented on macOS and Windows.

---

void **window_set_flag**(flag: WindowFlags, enabled: bool, window_id: int = 0)

Enables or disables the given window's given `flag`. See WindowFlags for possible values and their behavior.

---

void **window_set_ime_active**(active: bool, window_id: int = 0)

Sets whether Input Method Editor [https://en.wikipedia.org/wiki/Input_method] should be enabled for the window specified by `window_id`. See also window_set_ime_position.

---

void **window_set_ime_position**(position: Vector2i, window_id: int = 0)

Sets the position of the Input Method Editor [https://en.wikipedia.org/wiki/Input_method] popup for the specified `window_id`. Only effective if window_set_ime_active was set to `true` for the specified `window_id`.

---

void **window_set_input_event_callback**(callback: Callable, window_id: int = 0)

Sets the `callback` that should be called when any InputEvent is sent to the window specified by `window_id`.

Warning: Advanced users only! Adding such a callback to a Window node will override its default implementation, which can introduce bugs.

---

void **window_set_input_text_callback**(callback: Callable, window_id: int = 0)

Sets the `callback` that should be called when text is entered using the virtual keyboard to the window specified by `window_id`.

Warning: Advanced users only! Adding such a callback to a Window node will override its default implementation, which can introduce bugs.

---

void **window_set_max_size**(max_size: Vector2i, window_id: int = 0)

Sets the maximum size of the window specified by `window_id` in pixels. Normally, the user will not be able to drag the window to make it larger than the specified size. See also window_get_max_size.

Note: It's recommended to change this value using Window.max_size instead.

Note: Using third-party tools, it is possible for users to disable window geometry restrictions and therefore bypass this limit.

---

void **window_set_min_size**(min_size: Vector2i, window_id: int = 0)

Sets the minimum size for the given window to `min_size` in pixels. Normally, the user will not be able to drag the window to make it smaller than the specified size. See also window_get_min_size.

Note: It's recommended to change this value using Window.min_size instead.

Note: By default, the main window has a minimum size of `Vector2i(64, 64)`. This prevents issues that can arise when the window is resized to a near-zero size.

Note: Using third-party tools, it is possible for users to disable window geometry restrictions and therefore bypass this limit.

---

void **window_set_mode**(mode: WindowMode, window_id: int = 0) 🔗

Sets window mode for the given window to `mode`. See WindowMode for possible values and how each mode behaves.

Note: Setting the window to full screen forcibly sets the borderless flag to `true`, so make sure to set it back to `false` when not wanted.

---

void **window_set_mouse_passthrough**(region: PackedVector2Array, window_id: int = 0) 🔗

Sets a polygonal region of the window which accepts mouse events. Mouse events outside the region will be passed through.

Passing an empty array will disable passthrough support (all mouse events will be intercepted by the window, which is the default behavior).

GDScript

```
# Set region, using Path2D node.
DisplayServer.window_set_mouse_passthrough($Path2D.curve.get_baked_points())

# Set region, using Polygon2D node.
DisplayServer.window_set_mouse_passthrough($Polygon2D.polygon)

# Reset region to default.
DisplayServer.window_set_mouse_passthrough([])
```

C#

```
// Set region, using Path2D node.
DisplayServer.WindowSetMousePassthrough(GetNode<Path2D>
("Path2D").Curve.GetBakedPoints());

// Set region, using Polygon2D node.
DisplayServer.WindowSetMousePassthrough(GetNode<Polygon2D>
("Polygon2D").Polygon);

// Reset region to default.
DisplayServer.WindowSetMousePassthrough(new Vector2[] {});
```

Note: On Windows, the portion of a window that lies outside the region is not drawn, while on Linux (X11) and macOS it is.

Note: This method is implemented on Linux (X11), macOS and Windows.

---

void **window_set_popup_safe_rect**(window: int, rect: Rect2i)

Sets the bounding box of control, or menu item that was used to open the popup window, in the screen coordinate system. Clicking this area will not auto-close this popup.

---

void **window_set_position**(position: Vector2i, window_id: int = 0)

Sets the position of the given window to `position`. On multi-monitor setups, the screen position is relative to the virtual desktop area. On multi-monitor setups with different screen resolutions or orientations, the origin may be located outside any display like this:

```
* (0, 0)        +-------+
                |       |
+-------------+ |       |
|             | |       |
|             | |       |
+-------------+ +-------+
```

See also window_get_position and window_set_size.

Note: It's recommended to change this value using Window.position instead.

Note: On Linux (Wayland): this method is a no-op.

---

void **window_set_rect_changed_callback**(callback: Callable, window_id: int = 0)

Sets the `callback` that will be called when the window specified by `window_id` is moved or resized.

Warning: Advanced users only! Adding such a callback to a Window node will override its default implementation, which can introduce bugs.

void **window_set_size**(size: Vector2i, window_id: int = 0)

Sets the size of the given window to `size` (in pixels). See also window_get_size and window_get_position.

Note: It's recommended to change this value using Window.size instead.

void **window_set_title**(title: String, window_id: int = 0)

Sets the title of the given window to `title`.

Note: It's recommended to change this value using Window.title instead.

Note: Avoid changing the window title every frame, as this can cause performance issues on certain window managers. Try to change the window title only a few times per second at most.

void **window_set_transient**(window_id: int, parent_window_id: int)

Sets window transient parent. Transient window is will be destroyed with its transient parent and will return focus to their parent when closed. The transient window is displayed on top of a non-exclusive full-screen parent window. Transient windows can't enter full-screen mode.

Note: It's recommended to change this value using Window.transient instead.

Note: The behavior might be different depending on the platform.

void **window_set_vsync_mode**(vsync_mode: VSyncMode, window_id: int = 0)

Sets the V-Sync mode of the given window. See also ProjectSettings.display/window/vsync/vsync_mode.

See VSyncMode for possible values and how they affect the behavior of your application.

Depending on the platform and used renderer, the engine will fall back to VSYNC_ENABLED if the desired mode is not supported.

Note: V-Sync modes other than VSYNC_ENABLED are only supported in the Forward+ and Mobile rendering methods, not Compatibility.

---

void **window_set_window_buttons_offset**(offset: Vector2i, window_id: int = 0) 🔗

When WINDOW_FLAG_EXTEND_TO_TITLE flag is set, set offset to the center of the first titlebar button.

Note: This flag is implemented only on macOS.

---

void **window_set_window_event_callback**(callback: Callable, window_id: int = 0) 🔗

Sets the `callback` that will be called when an event occurs in the window specified by `window_id`.

Warning: Advanced users only! Adding such a callback to a Window node will override its default implementation, which can introduce bugs.

DTLSServer

Inherits: RefCounted < Object

Helper class to implement a DTLS server.

Description

This class is used to store the state of a DTLS server. Upon setup it converts connected PacketPeerUDP to PacketPeerDTLS accepting them via take_connection as DTLS clients. Under the hood, this class is used to store the DTLS state and cookies of the server. The reason of why the state and cookies are needed is outside of the scope of this documentation.

Below a small example of how to use it:

GDScript

```
# server_node.gd
extends Node

var dtls := DTLSServer.new()
var server := UDPServer.new()
var peers = []

func _ready():
    server.listen(4242)
    var key = load("key.key") # Your private key.
    var cert = load("cert.crt") # Your X509 certificate.
    dtls.setup(key, cert)

func _process(delta):
    while server.is_connection_available():
        var peer: PacketPeerUDP = server.take_connection()
        var dtls_peer: PacketPeerDTLS =
dtls.take_connection(peer)
        if dtls_peer.get_status() !=
PacketPeerDTLS.STATUS_HANDSHAKING:
            continue # It is normal that 50% of the connections
fails due to cookie exchange.
        print("Peer connected!")
        peers.append(dtls_peer)

    for p in peers:
        p.poll() # Must poll to update the state.
        if p.get_status() == PacketPeerDTLS.STATUS_CONNECTED:
            while p.get_available_packet_count() > 0:
                print("Received message from client: %s" %
p.get_packet().get_string_from_utf8())
                p.put_packet("Hello DTLS
client".to_utf8_buffer())
```

C#

```
// ServerNode.cs
using Godot;

public partial class ServerNode : Node
{
    private DtlsServer _dtls = new DtlsServer();
    private UdpServer _server = new UdpServer();
    private Godot.Collections.Array<PacketPeerDtls> _peers =
new Godot.Collections.Array<PacketPeerDtls>();

    public override void _Ready()
    {
        _server.Listen(4242);
        var key = GD.Load<CryptoKey>("key.key"); // Your
private key.
        var cert = GD.Load<X509Certificate>("cert.crt"); //
Your X509 certificate.
        _dtls.Setup(key, cert);
    }

    public override void _Process(double delta)
    {
        while (Server.IsConnectionAvailable())
        {
            PacketPeerUdp peer = _server.TakeConnection();
            PacketPeerDtls dtlsPeer =
_dtls.TakeConnection(peer);
            if (dtlsPeer.GetStatus() !=
PacketPeerDtls.Status.Handshaking)
            {
                continue; // It is normal that 50% of the
connections fails due to cookie exchange.
            }
            GD.Print("Peer connected!");
            _peers.Add(dtlsPeer);
        }

        foreach (var p in _peers)
        {
            p.Poll(); // Must poll to update the state.
            if (p.GetStatus() ==
PacketPeerDtls.Status.Connected)
            {
                while (p.GetAvailablePacketCount() > 0)
                {
                    GD.Print($"Received Message From Client:
{p.GetPacket().GetStringFromUtf8()}");
                    p.PutPacket("Hello DTLS
Client".ToUtf8Buffer());
                }
            }
        }
    }
}
```

GDScript

```
# client_node.gd
extends Node

var dtls := PacketPeerDTLS.new()
var udp := PacketPeerUDP.new()
var connected = false

func _ready():
    udp.connect_to_host("127.0.0.1", 4242)
    dtls.connect_to_peer(udp, false) # Use true in production
for certificate validation!

func _process(delta):
    dtls.poll()
    if dtls.get_status() == PacketPeerDTLS.STATUS_CONNECTED:
        if !connected:
            # Try to contact server
            dtls.put_packet("The answer is...
42!".to_utf8_buffer())
        while dtls.get_available_packet_count() > 0:
            print("Connected: %s" %
dtls.get_packet().get_string_from_utf8())
            connected = true
```

C#

```
// ClientNode.cs
using Godot;
using System.Text;

public partial class ClientNode : Node
{
    private PacketPeerDtls _dtls = new PacketPeerDtls();
    private PacketPeerUdp _udp = new PacketPeerUdp();
    private bool _connected = false;

    public override void _Ready()
    {
        _udp.ConnectToHost("127.0.0.1", 4242);
        _dtls.ConnectToPeer(_udp, validateCerts: false); // Use
true in production for certificate validation!
    }

    public override void _Process(double delta)
    {
        _dtls.Poll();
        if (_dtls.GetStatus() ==
PacketPeerDtls.Status.Connected)
        {
            if (!_connected)
            {
                // Try to contact server
                _dtls.PutPacket("The Answer
Is..42!".ToUtf8Buffer());
            }
            while (_dtls.GetAvailablePacketCount() > 0)
            {
                GD.Print($"Connected:
{_dtls.GetPacket().GetStringFromUtf8()}");
                _connected = true;
            }
        }
    }
}
```

Methods

Error	setup(server_options: TLSOptions)
PacketPeerDTLS	take_connection(udp_peer: PacketPeerUDP)

Method Descriptions

Error **setup**(server_options: TLSOptions) 🔗

Setup the DTLS server to use the given `server_options`. See TLSOptions.server.

PacketPeerDTLS **take_connection**(udp_peer: PacketPeerUDP) 🔗

Try to initiate the DTLS handshake with the given `udp_peer` which must be already connected (see PacketPeerUDP.connect_to_host).

Note: You must check that the state of the return PacketPeerUDP is PacketPeerDTLS.STATUS_HANDSHAKING, as it is normal that 50% of the new connections will be invalid due to cookie exchange.

EditorDebuggerPlugin

Inherits: RefCounted **<** Object

A base class to implement debugger plugins.

Description

EditorDebuggerPlugin provides functions related to the editor side of the debugger.

To interact with the debugger, an instance of this class must be added to the editor via EditorPlugin.add_debugger_plugin.

Once added, the _setup_session callback will be called for every EditorDebuggerSession available to the plugin, and when new ones are created (the sessions may be inactive during this stage).

You can retrieve the available EditorDebuggerSessions via get_sessions or get a specific one via get_session.

GDScript

```
@tool
extends EditorPlugin

class ExampleEditorDebugger extends EditorDebuggerPlugin:

    func _has_capture(prefix):
        # Return true if you wish to handle message with this prefix.
        return prefix == "my_plugin"

    func _capture(message, data, session_id):
        if message == "my_plugin:ping":
```

```
get_session(session_id).send_message("my_plugin:echo", data)

    func _setup_session(session_id):
        # Add a new tab in the debugger session UI containing a
label.
        var label = Label.new()
        label.name = "Example plugin"
        label.text = "Example plugin"
        var session = get_session(session_id)
        # Listens to the session started and stopped signals.
        session.started.connect(func (): print("Session
started"))
        session.stopped.connect(func (): print("Session
stopped"))
        session.add_session_tab(label)

var debugger = ExampleEditorDebugger.new()

func _enter_tree():
    add_debugger_plugin(debugger)

func _exit_tree():
    remove_debugger_plugin(debugger)
```

Methods

void	_breakpoint_set_in_tree(script: Script, line: int, enabled: bool) virtual
void	_breakpoints_cleared_in_tree() virtual
bool	_capture(message: String, data: Array, session_id: int) virtual

void	_goto_script_line(script: Script, line: int) virtual
bool	_has_capture(capture: String) virtual const
void	_setup_session(session_id: int) virtual
EditorDebuggerSession	get_session(id: int)
Array	get_sessions()

Method Descriptions

void **_breakpoint_set_in_tree**(script: Script, line: int, enabled: bool) virtual 🔗

Override this method to be notified when a breakpoint is set in the editor.

void **_breakpoints_cleared_in_tree**() virtual 🔗

Override this method to be notified when all breakpoints are cleared in the editor.

bool **_capture**(message: String, data: Array, session_id: int) virtual 🔗

Override this method to process incoming messages. The `session_id` is the ID of the EditorDebuggerSession that received the message (which you can retrieve via get_session).

---

void **_goto_script_line**(script: Script, line: int) virtual

Override this method to be notified when a breakpoint line has been clicked in the debugger breakpoint panel.

---

bool **_has_capture**(capture: String) virtual const

Override this method to enable receiving messages from the debugger. If `capture` is "my_message" then messages starting with "my_message:" will be passes to the _capture method.

---

void **_setup_session**(session_id: int) virtual

Override this method to be notified whenever a new EditorDebuggerSession is created (the session may be inactive during this stage).

EditorDebuggerSession **get_session**(id: int)

Returns the EditorDebuggerSession with the given `id`.

Array **get_sessions**()

Returns an array of EditorDebuggerSession currently available to this debugger plugin.

Note: Sessions in the array may be inactive, check their state via EditorDebuggerSession.is_active.

EditorDebuggerSession

Inherits: RefCounted < Object

A class to interact with the editor debugger.

Description

This class cannot be directly instantiated and must be retrieved via a EditorDebuggerPlugin.

You can add tabs to the session UI via add_session_tab, send messages via send_message, and toggle EngineProfilers via toggle_profiler.

Methods

void	add_session_tab(control: Control)
bool	is_active()
bool	is_breaked()
bool	is_debuggable()
void	remove_session_tab(control: Control)

void	send_message(message: String, data: Array = [])
void	set_breakpoint(path: String, line: int, enabled: bool)
void	toggle_profiler(profiler: String, enable: bool, data: Array = [])

Signals

breaked(can_debug: bool)

Emitted when the attached remote instance enters a break state. If `can_debug` is `true`, the remote instance will enter the debug loop.

continued()

Emitted when the attached remote instance exits a break state.

started()

Emitted when a remote instance is attached to this session (i.e. the session becomes active).

stopped()

Emitted when a remote instance is detached from this session (i.e. the session becomes inactive).

Method Descriptions

void **add_session_tab**(control: Control) 🔗

Adds the given `control` to the debug session UI in the debugger bottom panel.

---

bool **is_active**() 🔗

Returns `true` if the debug session is currently attached to a remote instance.

---

bool **is_breaked**() 🔗

Returns `true` if the attached remote instance is currently in the debug loop.

---

bool **is_debuggable**() 🔗

Returns `true` if the attached remote instance can be debugged.

---

void **remove_session_tab**(control: Control) 🔗

Removes the given `control` from the debug session UI in the debugger bottom panel.

---

void **send_message**(message: String, data: Array = []) 🔗

Sends the given `message` to the attached remote instance, optionally passing additionally `data`. See EngineDebugger for how to retrieve those messages.

---

void **set_breakpoint**(path: String, line: int, enabled: bool)

Enables or disables a specific breakpoint based on `enabled`, updating the Editor Breakpoint Panel accordingly.

---

void **toggle_profiler**(profiler: String, enable: bool, data: Array = [])

Toggle the given `profiler` on the attached remote instance, optionally passing additionally `data`. See EngineProfiler for more details.

EditorExportPlatform

Inherits: RefCounted **<** Object

Inherited By: EditorExportPlatformAndroid, EditorExportPlatformIOS, EditorExportPlatformMacOS, EditorExportPlatformPC, EditorExportPlatformWeb

Identifies a supported export platform, and internally provides the functionality of exporting to that platform.

Description

Base resource that provides the functionality of exporting a release build of a project to a platform, from the editor. Stores platform-specific metadata such as the name and supported features of the platform, and performs the exporting of projects, PCK files, and ZIP files. Uses an export template for the platform provided at the time of project exporting.

Used in scripting by EditorExportPlugin to configure platform-specific customization of scenes and resources. See EditorExportPlugin._begin_customize_scenes and EditorExportPlugin._begin_customize_resources for more details.

Tutorials

- Console support in Godot

Methods

String get_os_name() const

Method Descriptions

String **get_os_name**() const 🔗

Returns the name of the export operating system handled by this **EditorExportPlatform** class, as a friendly string. Possible return values are `Windows`, `Linux`, `macOS`, `Android`, `iOS`, and `Web`.

EditorExportPlatformAndroid

Inherits: EditorExportPlatform **<** RefCounted **<** Object

Exporter for Android.

Tutorials

- Exporting for Android
- Gradle builds for Android
- Android plugins documentation index

Properties

String	apk_expansion/SALT
bool	apk_expansion/enable
String	apk_expansion/public_key
bool	architectures/arm64-v8a
bool	architectures/armeabi-v7a
bool	architectures/x86

bool	architectures/x86_64
String	command_line/extra_args
String	custom_template/debug
String	custom_template/release
String	gradle_build/android_source_template
bool	gradle_build/compress_native_libraries
int	gradle_build/export_format
String	gradle_build/gradle_build_directory
String	gradle_build/min_sdk
String	gradle_build/target_sdk
bool	gradle_build/use_gradle_build
bool	graphics/opengl_debug
String	keystore/debug

String	keystore/debug_password
String	keystore/debug_user
String	keystore/release
String	keystore/release_password
String	keystore/release_user
String	launcher_icons/adaptive_background_432x432
String	launcher_icons/adaptive_foreground_432x432
String	launcher_icons/main_192x192
int	package/app_category
bool	package/exclude_from_recents
String	package/name
bool	package/retain_data_on_uninstall
bool	package/show_as_launcher_app

bool	package/show_in_android_tv
bool	package/show_in_app_library
bool	package/signed
String	package/unique_name
bool	permissions/access_checkin_properties
bool	permissions/access_coarse_location
bool	permissions/access_fine_location
bool	permissions/access_location_extra_commands
bool	permissions/access_mock_location
bool	permissions/access_network_state
bool	permissions/access_surface_flinger
bool	permissions/access_wifi_state
bool	permissions/account_manager

bool	permissions/add_voicemail
bool	permissions/authenticate_accounts
bool	permissions/battery_stats
bool	permissions/bind_accessibility_service
bool	permissions/bind_appwidget
bool	permissions/bind_device_admin
bool	permissions/bind_input_method
bool	permissions/bind_nfc_service
bool	permissions/bind_notification_listener_service
bool	permissions/bind_print_service
bool	permissions/bind_remoteviews
bool	permissions/bind_text_service

bool	permissions/bind_vpn_service
bool	permissions/bind_wallpaper
bool	permissions/bluetooth
bool	permissions/bluetooth_admin
bool	permissions/bluetooth_privileged
bool	permissions/brick
bool	permissions/broadcast_package_removed
bool	permissions/broadcast_sms
bool	permissions/broadcast_sticky
bool	permissions/broadcast_wap_push
bool	permissions/call_phone
bool	permissions/call_privileged
bool	permissions/camera

bool	permissions/capture_audio_output
bool	permissions/capture_secure_video_output
bool	permissions/capture_video_output
bool	permissions/change_component_enabled_state
bool	permissions/change_configuration
bool	permissions/change_network_state
bool	permissions/change_wifi_multicast_state
bool	permissions/change_wifi_state
bool	permissions/clear_app_cache
bool	permissions/clear_app_user_data
bool	permissions/control_location_updates
PackedStringArray	permissions/custom_permissions
bool	permissions/delete_cache_files

bool	permissions/delete_packages
bool	permissions/device_power
bool	permissions/diagnostic
bool	permissions/disable_keyguard
bool	permissions/dump
bool	permissions/expand_status_bar
bool	permissions/factory_test
bool	permissions/flashlight
bool	permissions/force_back
bool	permissions/get_accounts
bool	permissions/get_package_size
bool	permissions/get_tasks
bool	permissions/get_top_activity_info

bool	permissions/global_search
bool	permissions/hardware_test
bool	permissions/inject_events
bool	permissions/install_location_provider
bool	permissions/install_packages
bool	permissions/install_shortcut
bool	permissions/internal_system_window
bool	permissions/internet
bool	permissions/kill_background_processes
bool	permissions/location_hardware
bool	permissions/manage_accounts
bool	permissions/manage_app_tokens

bool	permissions/manage_documents
bool	permissions/manage_external_storage
bool	permissions/master_clear
bool	permissions/media_content_control
bool	permissions/modify_audio_settings
bool	permissions/modify_phone_state
bool	permissions/mount_format_filesystems
bool	permissions/mount_unmount_filesystems
bool	permissions/nfc
bool	permissions/persistent_activity
bool	permissions/post_notifications
bool	permissions/process_outgoing_calls
bool	permissions/read_calendar

bool	permissions/read_call_log
bool	permissions/read_contacts
bool	permissions/read_external_storage
bool	permissions/read_frame_buffer
bool	permissions/read_history_bookmarks
bool	permissions/read_input_state
bool	permissions/read_logs
bool	permissions/read_phone_state
bool	permissions/read_profile
bool	permissions/read_sms
bool	permissions/read_social_stream
bool	permissions/read_sync_settings
bool	permissions/read_sync_stats

bool	permissions/read_user_dictionary
bool	permissions/reboot
bool	permissions/receive_boot_completed
bool	permissions/receive_mms
bool	permissions/receive_sms
bool	permissions/receive_wap_push
bool	permissions/record_audio
bool	permissions/reorder_tasks
bool	permissions/restart_packages
bool	permissions/send_respond_via_message
bool	permissions/send_sms
bool	permissions/set_activity_watcher
bool	permissions/set_alarm

bool	permissions/set_always_finish
bool	permissions/set_animation_scale
bool	permissions/set_debug_app
bool	permissions/set_orientation
bool	permissions/set_pointer_speed
bool	permissions/set_preferred_applications
bool	permissions/set_process_limit
bool	permissions/set_time
bool	permissions/set_time_zone
bool	permissions/set_wallpaper
bool	permissions/set_wallpaper_hints
bool	permissions/signal_persistent_processes

bool	permissions/status_bar
bool	permissions/subscribed_feeds_read
bool	permissions/subscribed_feeds_write
bool	permissions/system_alert_window
bool	permissions/transmit_ir
bool	permissions/uninstall_shortcut
bool	permissions/update_device_stats
bool	permissions/use_credentials
bool	permissions/use_sip
bool	permissions/vibrate
bool	permissions/wake_lock
bool	permissions/write_apn_settings
bool	permissions/write_calendar

bool	permissions/write_call_log
bool	permissions/write_contacts
bool	permissions/write_external_storage
bool	permissions/write_gservices
bool	permissions/write_history_bookmarks
bool	permissions/write_profile
bool	permissions/write_secure_settings
bool	permissions/write_settings
bool	permissions/write_sms
bool	permissions/write_social_stream
bool	permissions/write_sync_settings
bool	permissions/write_user_dictionary
bool	screen/immersive_mode

bool	screen/support_large
bool	screen/support_normal
bool	screen/support_small
bool	screen/support_xlarge
bool	user_data_backup/allow
int	version/code
String	version/name
int	xr_features/xr_mode

Property Descriptions

String **apk_expansion/SALT** 🔗

Array of random bytes that the licensing Policy uses to create an Obfuscator [https://developer.android.com/google/play/licensing/adding-licensing#impl-Obfuscator].

bool **apk_expansion/enable** 🔗

If `true`, project resources are stored in the separate APK expansion file, instead of the APK.

Note: APK expansion should be enabled to use PCK encryption. See APK Expansion Files [https://developer.android.com/google/play/expansion-files]

String **apk_expansion/public_key**

Base64 encoded RSA public key for your publisher account, available from the profile page on the "Google Play Console".

---

bool **architectures/arm64-v8a**

If `true`, `arm64` binaries are included into exported project.

---

bool **architectures/armeabi-v7a**

If `true`, `arm32` binaries are included into exported project.

---

bool **architectures/x86**

If `true`, `x86_32` binaries are included into exported project.

---

bool **architectures/x86_64**

If `true`, `x86_64` binaries are included into exported project.

String **command_line/extra_args**

A list of additional command line arguments, separated by space, which the exported project will receive when started.

String **custom_template/debug** 🔗

Path to an APK file to use as a custom export template for debug exports. If left empty, default template is used.

Note: This is only used if gradle_build/use_gradle_build is disabled.

String **custom_template/release** 🔗

Path to an APK file to use as a custom export template for release exports. If left empty, default template is used.

Note: This is only used if gradle_build/use_gradle_build is disabled.

String **gradle_build/android_source_template** 🔗

Path to a ZIP file holding the source for the export template used in a Gradle build. If left empty, the default template is used.

bool **gradle_build/compress_native_libraries** 🔗

If `true`, native libraries are compressed when performing a Gradle build.

Note: Although your binary may be smaller, your application may load slower because the native libraries are not loaded directly from the binary at runtime.

int **gradle_build/export_format** 🔗

Application export format (*.apk or *.aab).

String **gradle_build/gradle_build_directory**

Path to the Gradle build directory. If left empty, then `res://android` will be used.

String **gradle_build/min_sdk**

Minimum Android API level required for the application to run (used during Gradle build). See android:minSdkVersion [https://developer.android.com/guide/topics/manifest/uses-sdk-element#uses].

String **gradle_build/target_sdk**

The Android API level on which the application is designed to run (used during Gradle build). See android:targetSdkVersion [https://developer.android.com/guide/topics/manifest/uses-sdk-element#uses].

---

bool **gradle_build/use_gradle_build**

If `true`, Gradle build is used instead of pre-built APK.

---

bool **graphics/opengl_debug**

If `true`, OpenGL ES debug context will be created (additional runtime checking, validation, and logging).

String **keystore/debug**

Path of the debug keystore file.

Can be overridden with the environment variable `GODOT_ANDROID_KEYSTORE_DEBUG_PATH`.

Fallbacks to `EditorSettings.export/android/debug_keystore` if empty.

String **keystore/debug_password** 🔗

Password for the debug keystore file.

Can be overridden with the environment variable `GODOT_ANDROID_KEYSTORE_DEBUG_PASSWORD`.

Fallbacks to `EditorSettings.export/android/debug_keystore_pass` if both it and keystore/debug are empty.

String **keystore/debug_user** 🔗

User name for the debug keystore file.

Can be overridden with the environment variable `GODOT_ANDROID_KEYSTORE_DEBUG_USER`.

Fallbacks to `EditorSettings.export/android/debug_keystore_user` if both it and keystore/debug are empty.

String **keystore/release** 🔗

Path of the release keystore file.

Can be overridden with the environment variable `GODOT_ANDROID_KEYSTORE_RELEASE_PATH`.

String **keystore/release_password** 🔗

Password for the release keystore file.

Can be overridden with the environment variable `GODOT_ANDROID_KEYSTORE_RELEASE_PASSWORD`.

String **keystore/release_user** 🔗

User name for the release keystore file.

Can be overridden with the environment variable `GODOT_ANDROID_KEYSTORE_RELEASE_USER`.

String **launcher_icons/adaptive_background_432x432** 🔗

Background layer of the application adaptive icon file. See Design adaptive icons [https://developer.android.com/develop/ui/views/launch/icon_design_adaptive#design-adaptive-icons].

String **launcher_icons/adaptive_foreground_432x432** 🔗

Foreground layer of the application adaptive icon file. See Design adaptive icons [https://developer.android.com/develop/ui/views/launch/icon_design_adaptive#design-adaptive-icons].

String **launcher_icons/main_192x192** 🔗

Application icon file. If left empty, it will fallback to ProjectSettings.application/config/icon.

---

int **package/app_category**

Application category for the Google Play Store. Only define this if your application fits one of the categories well. See android:appCategory [https://developer.android.com/guide/topics/manifest/application-element#appCategory].

---

bool **package/exclude_from_recents**

If `true`, task initiated by main activity will be excluded from the list of recently used applications. See android:excludeFromRecents [https://developer.android.com/guide/topics/manifest/activity-element#exclude].

String **package/name**

Name of the application.

---

bool **package/retain_data_on_uninstall**

If `true`, when the user uninstalls an app, a prompt to keep the app's data will be shown. See android:hasFragileUserData [https://developer.android.com/guide/topics/manifest/application-element#fragileuserdata].

---

bool **package/show_as_launcher_app**

If `true`, the user will be able to set this app as the system launcher in Android preferences.

---

bool **package/show_in_android_tv**

If `true`, this app will show in Android TV launcher UI.

---

bool **package/show_in_app_library**

If `true`, this app will show in the device's app library.

Note: This is `true` by default.

---

bool **package/signed**

If `true`, package signing is enabled.

String **package/unique_name**

Unique application identifier in a reverse-DNS format. The reverse DNS format should preferably match a domain name you control, but this is not strictly required. For instance, if you own `example.com`, your package unique name should preferably be of the form `com.example.mygame`. This identifier can only contain lowercase alphanumeric characters (`a-z`, and `0-9`), underscores (`_`), and periods (`.`). Each component of the reverse DNS format must start with a letter: for instance, `com.example.8game` is not valid.

If `$genname` is present in the value, it will be replaced by the project name converted to lowercase. If there are invalid characters in the project name, they will be stripped. If all characters in the project name are stripped, `$genname` is replaced by `noname`.

Note: Changing the package name will cause the package to be considered as a new package, with its own installation and data paths. The new package won't be usable to update existing installations.

Note: When publishing to Google Play, the package name must be *globally* unique. This means no other apps published on Google Play must be using the same package name as yours. Otherwise, you'll be prevented from publishing your app on Google Play.

bool **permissions/access_checkin_properties**

Allows read/write access to the "properties" table in the checkin database. See ACCESS_CHECKIN_PROPERTIES [https://developer.android.com/reference/android/Manifest.permission#ACCESS_CHECKIN_PROPERTIES].

bool **permissions/access_coarse_location**

Allows access to the approximate location information. See ACCESS_COARSE_LOCATION [https://developer.android.com/reference/android/Manifest.permission#ACCESS_COARSE_LOCATION].

bool **permissions/access_fine_location**

Allows access to the precise location information. See ACCESS_FINE_LOCATION [https://developer.android.com/reference/android/Manifest.permission#ACCESS_FINE_LOCATION].

bool **permissions/access_location_extra_commands**

Allows access to the extra location provider commands. See ACCESS_LOCATION_EXTRA_COMMANDS [https://developer.android.com/reference/android/Manifest.permission#ACCESS_LOCATION_EXTRA_COMMANDS].

---

bool **permissions/access_mock_location**

Allows an application to create mock location providers for testing.

---

bool **permissions/access_network_state**

Allows access to the information about networks. See ACCESS_NETWORK_STATE [https://developer.android.com/reference/android/Manifest.permission#ACCESS_NETWORK_STATE].

---

bool **permissions/access_surface_flinger**

Allows an application to use SurfaceFlinger's low level features.

---

bool **permissions/access_wifi_state**

Allows access to the information about Wi-Fi networks. See ACCESS_WIFI_STATE [https://developer.android.com/reference/android/Manifest.permission#ACCESS_WIFI_STATE].

---

bool **permissions/account_manager**

Allows applications to call into AccountAuthenticators. See ACCOUNT_MANAGER [https://developer.android.com/reference/android/Manifest.permission#ACCOUNT_MANAGER].

---

bool **permissions/add_voicemail**

Allows an application to add voicemails into the system. See ADD_VOICEMAIL [https://developer.android.com/reference/android/Manifest.permission#ADD_VOICEMAIL].

---

bool **permissions/authenticate_accounts** 🔗

Allows an application to act as an AccountAuthenticator for the AccountManager.

---

bool **permissions/battery_stats** 🔗

Allows an application to collect battery statistics. See BATTERY_STATS [https://developer.android.com/reference/android/Manifest.permission#BATTERY_STATS].

---

bool **permissions/bind_accessibility_service** 🔗

Must be required by an AccessibilityService, to ensure that only the system can bind to it. See BIND_ACCESSIBILITY_SERVICE [https://developer.android.com/reference/android/Manifest.permission#BIND_ACCESSIBILITY_SERVICE].

---

bool **permissions/bind_appwidget** 🔗

Allows an application to tell the AppWidget service which application can access AppWidget's data. See BIND_APPWIDGET [https://developer.android.com/reference/android/Manifest.permission#BIND_APPWIDGET].

---

bool **permissions/bind_device_admin**

Must be required by device administration receiver, to ensure that only the system can interact with it. See BIND_DEVICE_ADMIN [https://developer.android.com/reference/android/Manifest.permission#BIND_DEVICE_ADMIN].

---

bool **permissions/bind_input_method**

Must be required by an InputMethodService, to ensure that only the system can bind to it. See BIND_INPUT_METHOD [https://developer.android.com/reference/android/Manifest.permission#BIND_INPUT_METHOD].

---

bool **permissions/bind_nfc_service**

Must be required by a HostApduService or OffHostApduService to ensure that only the system can bind to it. See BIND_NFC_SERVICE [https://developer.android.com/reference/android/Manifest.permission#BIND_NFC_SERVICE].

---

bool **permissions/bind_notification_listener_service**

Must be required by a NotificationListenerService, to ensure that only the system can bind to it. See BIND_NOTIFICATION_LISTENER_SERVICE [https://developer.android.com/reference/android/Manifest.permission#BIND_NOTIFICATION_LISTENER_SERVICE].

---

bool **permissions/bind_print_service**

Must be required by a PrintService, to ensure that only the system can bind to it. See BIND_PRINT_SERVICE [https://developer.android.com/reference/android/Manifest.permission#BIND_PRINT_SERVICE].

---

bool **permissions/bind_remoteviews** 🔗

Must be required by a RemoteViewsService, to ensure that only the system can bind to it. See BIND_REMOTEVIEWS [https://developer.android.com/reference/android/Manifest.permission#BIND_REMOTEVIEWS].

---

bool **permissions/bind_text_service** 🔗

Must be required by a TextService (e.g. SpellCheckerService) to ensure that only the system can bind to it. See BIND_TEXT_SERVICE [https://developer.android.com/reference/android/Manifest.permission#BIND_TEXT_SERVICE].

---

bool **permissions/bind_vpn_service** 🔗

Must be required by a VpnService, to ensure that only the system can bind to it. See BIND_VPN_SERVICE [https://developer.android.com/reference/android/Manifest.permission#BIND_VPN_SERVICE].

---

bool **permissions/bind_wallpaper** 🔗

Must be required by a WallpaperService, to ensure that only the system can bind to it. See BIND_WALLPAPER [https://developer.android.com/reference/android/Manifest.permission#BIND_WALLPAPER].

---

bool **permissions/bluetooth** 🔗

Allows applications to connect to paired bluetooth devices. See BLUETOOTH [https://developer.android.com/reference/android/Manifest.permission#BLUETOOTH].

---

bool **permissions/bluetooth_admin** 🔗

Allows applications to discover and pair bluetooth devices. See BLUETOOTH_ADMIN [https://developer.android.com/reference/android/Manifest.permission#BLUETOOTH_ADMIN].

---

bool **permissions/bluetooth_privileged** 🔗

Allows applications to pair bluetooth devices without user interaction, and to allow or disallow phonebook access or message access. See BLUETOOTH_PRIVILEGED [https://developer.android.com/reference/android/Manifest.permission#BLUETOOTH_PRIVILEGED].

---

bool **permissions/brick** 🔗

Required to be able to disable the device (very dangerous!).

---

bool **permissions/broadcast_package_removed** 🔗

Allows an application to broadcast a notification that an application package has been removed. See BROADCAST_PACKAGE_REMOVED [https://developer.android.com/reference/android/Manifest.permission#BROADCAST_PACKAGE_REMOVED].

bool **permissions/broadcast_sms**

Allows an application to broadcast an SMS receipt notification. See BROADCAST_SMS [https://developer.android.com/reference/android/Manifest.permission#BROADCAST_SMS].

bool **permissions/broadcast_sticky**

Allows an application to broadcast sticky intents. See BROADCAST_STICKY [https://developer.android.com/reference/android/Manifest.permission#BROADCAST_STICKY].

bool **permissions/broadcast_wap_push**

Allows an application to broadcast a WAP PUSH receipt notification. See BROADCAST_WAP_PUSH [https://developer.android.com/reference/android/Manifest.permission#BROADCAST_WAP_PUSH].

bool **permissions/call_phone**

Allows an application to initiate a phone call without going through the Dialer user interface. See CALL_PHONE [https://developer.android.com/reference/android/Manifest.permission#CALL_PHONE].

---

bool **permissions/call_privileged**

Allows an application to call any phone number, including emergency numbers, without going through the Dialer user interface. See CALL_PRIVILEGED [https://developer.android.com/reference/android/Manifest.permission#CALL_PRIVILEGED].

---

bool **permissions/camera**

Required to be able to access the camera device. See CAMERA [https://developer.android.com/reference/android/Manifest.permission#CAMERA].

---

bool **permissions/capture_audio_output**

Allows an application to capture audio output. See CAPTURE_AUDIO_OUTPUT [https://developer.android.com/reference/android/Manifest.permission#CAPTURE_AUDIO_OUTPUT].

---

bool **permissions/capture_secure_video_output**

Allows an application to capture secure video output.

---

bool **permissions/capture_video_output**

Allows an application to capture video output.

---

bool **permissions/change_component_enabled_state**

Allows an application to change whether an application component (other than its own) is enabled or not. See CHANGE_COMPONENT_ENABLED_STATE [https://developer.android.com/reference/android/Manifest.permission#CHANGE_COMPONENT_ENABLED_STATE].

---

bool **permissions/change_configuration** 🔗

Allows an application to modify the current configuration, such as locale. See CHANGE_CONFIGURATION [https://developer.android.com/reference/android/Manifest.permission#CHANGE_CONFIGURATION].

---

bool **permissions/change_network_state** 🔗

Allows applications to change network connectivity state. See CHANGE_NETWORK_STATE [https://developer.android.com/reference/android/Manifest.permission#CHANGE_NETWORK_STATE].

---

bool **permissions/change_wifi_multicast_state** 🔗

Allows applications to enter Wi-Fi Multicast mode. See CHANGE_WIFI_MULTICAST_STATE [https://developer.android.com/reference/android/Manifest.permission#CHANGE_WIFI_MULTICAST_STATE].

---

bool **permissions/change_wifi_state** 🔗

Allows applications to change Wi-Fi connectivity state. See CHANGE_WIFI_STATE [https://developer.android.com/reference/android/Manifest.permission#CHANGE_WIFI_STATE].

---

bool **permissions/clear_app_cache**

Allows an application to clear the caches of all installed applications on the device. See CLEAR_APP_CACHE [https://developer.android.com/reference/android/Manifest.permission#CLEAR_APP_CACHE].

---

bool **permissions/clear_app_user_data**

Allows an application to clear user data.

---

bool **permissions/control_location_updates**

Allows enabling/disabling location update notifications from the radio. See CONTROL_LOCATION_UPDATES [https://developer.android.com/reference/android/Manifest.permission#CONTROL_LOCATION_UPDATES].

PackedStringArray **permissions/custom_permissions**

Array of custom permission strings.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedStringArray for more details.

---

bool **permissions/delete_cache_files**

Deprecated: This property may be changed or removed in future versions.

---

bool **permissions/delete_packages** 🔗

Allows an application to delete packages. See DELETE_PACKAGES [https://developer.android.com/reference/android/Manifest.permission#DELETE_PACKAGES].

---

bool **permissions/device_power** 🔗

Allows low-level access to power management.

---

bool **permissions/diagnostic** 🔗

Allows applications to RW to diagnostic resources. See DIAGNOSTIC [https://developer.android.com/reference/android/Manifest.permission#DIAGNOSTIC].

---

bool **permissions/disable_keyguard** 🔗

Allows applications to disable the keyguard if it is not secure. See DISABLE_KEYGUARD [https://developer.android.com/reference/android/Manifest.permission#DISABLE_KEYGUARD].

---

bool **permissions/dump** 🔗

Allows an application to retrieve state dump information from system services. See DUMP [https://developer.android.com/reference/android/Manifest.permission#DUMP].

---

bool **permissions/expand_status_bar** 🔗

Allows an application to expand or collapse the status bar. See EXPAND_STATUS_BAR [https://developer.android.com/reference/android/Manifest.permission#EXPAND_STATUS_BAR].

---

bool **permissions/factory_test** 🔗

Run as a manufacturer test application, running as the root user. See FACTORY_TEST [https://developer.android.com/reference/android/Manifest.permission#FACTORY_TEST].

---

bool **permissions/flashlight** 🔗

Allows access to the flashlight.

---

bool **permissions/force_back** 🔗

Allows an application to force a BACK operation on whatever is the top activity.

---

bool **permissions/get_accounts** 🔗

Allows access to the list of accounts in the Accounts Service. See GET_ACCOUNTS [https://developer.android.com/reference/android/Manifest.permission#GET_ACCOUNTS].

---

bool **permissions/get_package_size** 🔗

Allows an application to find out the space used by any package. See GET_PACKAGE_SIZE [https://developer.android.com/reference/android/Manifest.permission#GET_PACKAGE_SIZE].

bool **permissions/get_tasks** 🔗

Deprecated: Deprecated in API level 21.

bool **permissions/get_top_activity_info** 🔗

Allows an application to retrieve private information about the current top activity.

bool **permissions/global_search** 🔗

Used on content providers to allow the global search system to access their data. See GLOBAL_SEARCH [https://developer.android.com/reference/android/Manifest.permission#GLOBAL_SEARCH].

bool **permissions/hardware_test** 🔗

Allows access to hardware peripherals.

bool **permissions/inject_events** 🔗

Allows an application to inject user events (keys, touch, trackball) into the event stream and deliver them to ANY window.

bool **permissions/install_location_provider** 🔗

Allows an application to install a location provider into the Location Manager. See INSTALL_LOCATION_PROVIDER [https://developer.android.com/reference/android/Manifest.permission#INSTALL_LOCATION_PROVIDER].

---

bool **permissions/install_packages** 🔗

Allows an application to install packages. See INSTALL_PACKAGES [https://developer.android.com/reference/android/Manifest.permission#INSTALL_PACKAGES].

---

bool **permissions/install_shortcut** 🔗

Allows an application to install a shortcut in Launcher. See INSTALL_SHORTCUT [https://developer.android.com/reference/android/Manifest.permission#INSTALL_SHORTCUT].

---

bool **permissions/internal_system_window** 🔗

Allows an application to open windows that are for use by parts of the system user interface.

---

bool **permissions/internet** 🔗

Allows applications to open network sockets. See INTERNET [https://developer.android.com/reference/android/Manifest.permission#INTERNET].

---

bool **permissions/kill_background_processes** 🔗

Allows an application to call ActivityManager.killBackgroundProcesses(String). See KILL_BACKGROUND_PROCESSES [https://developer.android.com/reference/android/Manifest.permission#KILL_BACKGROUND_PROCESSES].

---

bool **permissions/location_hardware**

Allows an application to use location features in hardware, such as the geofencing api. See LOCATION_HARDWARE [https://developer.android.com/reference/android/Manifest.permission#LOCATION_HARDWARE].

---

bool **permissions/manage_accounts**

Allows an application to manage the list of accounts in the AccountManager.

---

bool **permissions/manage_app_tokens**

Allows an application to manage (create, destroy, Z-order) application tokens in the window manager.

---

bool **permissions/manage_documents**

Allows an application to manage access to documents, usually as part of a document picker. See MANAGE_DOCUMENTS [https://developer.android.com/reference/android/Manifest.permission#MANAGE_DOCUMENTS].

---

bool **permissions/manage_external_storage**

Allows an application a broad access to external storage in scoped storage. See MANAGE_EXTERNAL_STORAGE [https://developer.android.com/reference/android/Manifest.permission#MANAGE_EXTERNAL_STORAGE].

---

bool **permissions/master_clear** 🔗

See MASTER_CLEAR [https://developer.android.com/reference/android/Manifest.permission#MASTER_CLEAR].

---

bool **permissions/media_content_control** 🔗

Allows an application to know what content is playing and control its playback. See MEDIA_CONTENT_CONTROL [https://developer.android.com/reference/android/Manifest.permission#MEDIA_CONTENT_CONTROL].

---

bool **permissions/modify_audio_settings** 🔗

Allows an application to modify global audio settings. See MODIFY_AUDIO_SETTINGS [https://developer.android.com/reference/android/Manifest.permission#MODIFY_AUDIO_SETTINGS].

---

bool **permissions/modify_phone_state** 🔗

Allows modification of the telephony state - power on, mmi, etc. Does not include placing calls. See MODIFY_PHONE_STATE [https://developer.android.com/reference/android/Manifest.permission#MODIFY_PHONE_STATE].

---

bool **permissions/mount_format_filesystems** 🔗

Allows formatting file systems for removable storage. See MOUNT_FORMAT_FILESYSTEMS [https://developer.android.com/reference/android/Manifest.permission#MOUNT_FORMAT_FILESYSTEMS].

---

bool **permissions/mount_unmount_filesystems** 🔗

Allows mounting and unmounting file systems for removable storage. See MOUNT_UNMOUNT_FILESYSTEMS [https://developer.android.com/reference/android/Manifest.permission#MOUNT_UNMOUNT_FILESYSTEMS].

---

bool **permissions/nfc** 🔗

Allows applications to perform I/O operations over NFC. See NFC [https://developer.android.com/reference/android/Manifest.permission#NFC].

---

bool **permissions/persistent_activity** 🔗

Deprecated: Deprecated in API level 15.

Allows an application to make its activities persistent.

---

bool **permissions/post_notifications** 🔗

Allows an application to post notifications. Added in API level 33. See Notification runtime permission [https://developer.android.com/develop/ui/views/notifications/notification-permission].

---

bool **permissions/process_outgoing_calls** 🔗

Deprecated: Deprecated in API level 29.

Allows an application to see the number being dialed during an outgoing call with the option to redirect the call to a different number or abort the call altogether. See PROCESS_OUTGOING_CALLS [https://developer.android.com/reference/android/Manifest.permission#PROCESS_OUTGOING_CALLS].

bool **permissions/read_calendar** 🔗

Allows an application to read the user's calendar data. See READ_CALENDAR [https://developer.android.com/reference/android/Manifest.permission#READ_CALENDAR].

bool **permissions/read_call_log** 🔗

Allows an application to read the user's call log. See READ_CALL_LOG [https://developer.android.com/reference/android/Manifest.permission#READ_CALL_LOG].

bool **permissions/read_contacts** 🔗

Allows an application to read the user's contacts data. See READ_CONTACTS [https://developer.android.com/reference/android/Manifest.permission#READ_CONTACTS].

bool **permissions/read_external_storage** 🔗

Deprecated: Deprecated in API level 33.

Allows an application to read from external storage. See READ_EXTERNAL_STORAGE [https://developer.android.com/reference/android/Manifest.permission#READ_EXTERNAL_STORAGE].

bool **permissions/read_frame_buffer** 🔗

Allows an application to take screen shots and more generally get access to the frame buffer data.

bool **permissions/read_history_bookmarks** 🔗

Allows an application to read (but not write) the user's browsing history and bookmarks.

bool **permissions/read_input_state** 🔗

Deprecated: Deprecated in API level 16.

bool **permissions/read_logs** 🔗

Allows an application to read the low-level system log files. See READ_LOGS [https://developer.android.com/reference/android/Manifest.permission#READ_LOGS].

bool **permissions/read_phone_state** 🔗

Allows read only access to phone state. See READ_PHONE_STATE [https://developer.android.com/reference/android/Manifest.permission#READ_PHONE_STATE].

bool **permissions/read_profile** 🔗

Allows an application to read the user's personal profile data.

bool **permissions/read_sms** 🔗

Allows an application to read SMS messages. See READ_SMS [https://developer.android.com/reference/android/Manifest.permission#READ_SMS].

bool **permissions/read_social_stream** 🔗

Allows an application to read from the user's social stream.

bool **permissions/read_sync_settings** 🔗

Allows applications to read the sync settings. See READ_SYNC_SETTINGS [https://developer.android.com/reference/android/Manifest.permission#READ_SYNC_SETTINGS].

bool **permissions/read_sync_stats** 🔗

Allows applications to read the sync stats. See READ_SYNC_STATS [https://developer.android.com/reference/android/Manifest.permission#READ_SYNC_STATS].

bool **permissions/read_user_dictionary**

Allows an application to read the user dictionary.

bool **permissions/reboot**

Required to be able to reboot the device. See REBOOT [https://developer.android.com/reference/android/Manifest.permission#REBOOT].

bool **permissions/receive_boot_completed**

Allows an application to receive the Intent.ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting. See RECEIVE_BOOT_COMPLETED [https://developer.android.com/reference/android/Manifest.permission#RECEIVE_BOOT_COMPLETED].

bool **permissions/receive_mms**

Allows an application to monitor incoming MMS messages. See RECEIVE_MMS [https://developer.android.com/reference/android/Manifest.permission#RECEIVE_MMS].

bool **permissions/receive_sms**

Allows an application to receive SMS messages. See RECEIVE_SMS [https://developer.android.com/reference/android/Manifest.permission#RECEIVE_SMS].

bool **permissions/receive_wap_push**

Allows an application to receive WAP push messages. See RECEIVE_WAP_PUSH [https://developer.android.com/reference/android/Manifest.permission#RECEIVE_WAP_PUSH].

---

bool **permissions/record_audio** 🔗

Allows an application to record audio. See RECORD_AUDIO [https://developer.android.com/reference/android/Manifest.permission#RECORD_AUDIO].

---

bool **permissions/reorder_tasks** 🔗

Allows an application to change the Z-order of tasks. See REORDER_TASKS [https://developer.android.com/reference/android/Manifest.permission#REORDER_TASKS].

---

bool **permissions/restart_packages** 🔗

Deprecated: Deprecated in API level 15.

---

bool **permissions/send_respond_via_message** 🔗

Allows an application (Phone) to send a request to other applications to handle the respond-via-message action during incoming calls. See SEND_RESPOND_VIA_MESSAGE [https://developer.android.com/reference/android/Manifest.permission#SEND_RESPOND_VIA_MESSAGE].

---

bool **permissions/send_sms** 🔗

Allows an application to send SMS messages. See SEND_SMS [https://developer.android.com/reference/android/Manifest.permission#SEND_SMS].

bool **permissions/set_activity_watcher**

Allows an application to watch and control how activities are started globally in the system.

bool **permissions/set_alarm**

Allows an application to broadcast an Intent to set an alarm for the user. See SET_ALARM [https://developer.android.com/reference/android/Manifest.permission#SET_ALARM].

bool **permissions/set_always_finish**

Allows an application to control whether activities are immediately finished when put in the background. See SET_ALWAYS_FINISH [https://developer.android.com/reference/android/Manifest.permission#SET_ALWAYS_FINISH].

bool **permissions/set_animation_scale**

Allows to modify the global animation scaling factor. See SET_ANIMATION_SCALE [https://developer.android.com/reference/android/Manifest.permission#SET_ANIMATION_SCALE].

bool **permissions/set_debug_app**

Configure an application for debugging. See SET_DEBUG_APP [https://developer.android.com/reference/android/Manifest.permission#SET_DEBUG_APP].

---

bool **permissions/set_orientation**

Allows low-level access to setting the orientation (actually rotation) of the screen.

---

bool **permissions/set_pointer_speed**

Allows low-level access to setting the pointer speed.

---

bool **permissions/set_preferred_applications**

Deprecated: Deprecated in API level 15.

---

bool **permissions/set_process_limit**

Allows an application to set the maximum number of (not needed) application processes that can be running. See SET_PROCESS_LIMIT [https://developer.android.com/reference/android/Manifest.permission#SET_PROCESS_LIMIT].

---

bool **permissions/set_time**

Allows applications to set the system time directly. See SET_TIME [https://developer.android.com/reference/android/Manifest.permission#SET_TIME].

---

bool **permissions/set_time_zone**

Allows applications to set the system time zone directly. See SET_TIME_ZONE [https://developer.android.com/reference/android/Manifest.permission#SET_TIME_ZONE].

---

bool **permissions/set_wallpaper**

Allows applications to set the wallpaper. See SET_WALLPAPER [https://developer.android.com/reference/android/Manifest.permission#SET_WALLPAPER].

---

bool **permissions/set_wallpaper_hints**

Allows applications to set the wallpaper hints. See SET_WALLPAPER_HINTS [https://developer.android.com/reference/android/Manifest.permission#SET_WALLPAPER_HINTS].

---

bool **permissions/signal_persistent_processes**

Allow an application to request that a signal be sent to all persistent processes. See SIGNAL_PERSISTENT_PROCESSES [https://developer.android.com/reference/android/Manifest.permission#SIGNAL_PERSISTENT_PROCESSES].

---

bool **permissions/status_bar**

Allows an application to open, close, or disable the status bar and its icons. See STATUS_BAR [https://developer.android.com/reference/android/Manifest.permission#STATUS_BAR].

---

bool **permissions/subscribed_feeds_read** 🔗

Allows an application to allow access the subscribed feeds ContentProvider.

bool **permissions/subscribed_feeds_write** 🔗

Deprecated: This property may be changed or removed in future versions.

bool **permissions/system_alert_window** 🔗

Allows an app to create windows using the type WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY, shown on top of all other apps. See SYSTEM_ALERT_WINDOW [https://developer.android.com/reference/android/Manifest.permission#SYSTEM_ALERT_WINDOW].

bool **permissions/transmit_ir** 🔗

Allows using the device's IR transmitter, if available. See TRANSMIT_IR [https://developer.android.com/reference/android/Manifest.permission#TRANSMIT_IR].

bool **permissions/uninstall_shortcut** 🔗

Deprecated: This property may be changed or removed in future versions.

bool **permissions/update_device_stats** 🔗

Allows an application to update device statistics. See UPDATE_DEVICE_STATS [https://developer.android.com/reference/android/Manifest.permission#UPDATE_DEVICE_STATS].

bool **permissions/use_credentials**

Allows an application to request authtokens from the AccountManager.

bool **permissions/use_sip**

Allows an application to use SIP service. See USE_SIP [https://developer.android.com/reference/android/Manifest.permission#USE_SIP].

bool **permissions/vibrate**

Allows access to the vibrator. See VIBRATE [https://developer.android.com/reference/android/Manifest.permission#VIBRATE].

bool **permissions/wake_lock**

Allows using PowerManager WakeLocks to keep processor from sleeping or screen from dimming. See WAKE_LOCK [https://developer.android.com/reference/android/Manifest.permission#WAKE_LOCK].

bool **permissions/write_apn_settings**

Allows applications to write the apn settings and read sensitive fields of an existing apn settings like user and password. See WRITE_APN_SETTINGS [https://developer.android.com/reference/android/Manifest.permission#WRITE_APN_SETTINGS].

---

bool **permissions/write_calendar**

Allows an application to write the user's calendar data. See WRITE_CALENDAR [https://developer.android.com/reference/android/Manifest.permission#WRITE_CALENDAR].

---

bool **permissions/write_call_log**

Allows an application to write (but not read) the user's call log data. See WRITE_CALL_LOG [https://developer.android.com/reference/android/Manifest.permission#WRITE_CALL_LOG].

---

bool **permissions/write_contacts**

Allows an application to write the user's contacts data. See WRITE_CONTACTS [https://developer.android.com/reference/android/Manifest.permission#WRITE_CONTACTS].

---

bool **permissions/write_external_storage**

Allows an application to write to external storage. See WRITE_EXTERNAL_STORAGE [https://developer.android.com/reference/android/Manifest.permission#WRITE_EXTERNAL_STORAGE].

---

bool **permissions/write_gservices**

Allows an application to modify the Google service map. See WRITE_GSERVICES [https://developer.android.com/reference/android/Manifest.permission#WRITE_GSERVICES].

bool **permissions/write_history_bookmarks**

Allows an application to write (but not read) the user's browsing history and bookmarks.

bool **permissions/write_profile**

Allows an application to write (but not read) the user's personal profile data.

bool **permissions/write_secure_settings**

Allows an application to read or write the secure system settings. See WRITE_SECURE_SETTINGS [https://developer.android.com/reference/android/Manifest.permission#WRITE_SECURE_SETTINGS].

bool **permissions/write_settings**

Allows an application to read or write the system settings. See WRITE_SETTINGS [https://developer.android.com/reference/android/Manifest.permission#WRITE_SETTINGS].

bool **permissions/write_sms** 🔗

Allows an application to write SMS messages.

bool **permissions/write_social_stream** 🔗

Allows an application to write (but not read) the user's social stream data.

bool **permissions/write_sync_settings** 🔗

Allows applications to write the sync settings. See WRITE_SYNC_SETTINGS [https://developer.android.com/reference/android/Manifest.permission#WRITE_SYNC_SETTINGS].

bool **permissions/write_user_dictionary** 🔗

Allows an application to write to the user dictionary.

bool **screen/immersive_mode** 🔗

If `true`, hides navigation and status bar.

bool **screen/support_large** 🔗

Indicates whether the application supports larger screen form-factors.

bool **screen/support_normal** 🔗

Indicates whether an application supports the "normal" screen form-factors.

---

bool **screen/support_small** 🔗

Indicates whether the application supports smaller screen form-factors.

---

bool **screen/support_xlarge** 🔗

Indicates whether the application supports extra large screen form-factors.

---

bool **user_data_backup/allow** 🔗

If `true`, allows the application to participate in the backup and restore infrastructure.

---

int **version/code** 🔗

Machine-readable application version. This must be incremented for every new release pushed to the Play Store.

String **version/name** 🔗

Application version visible to the user. Falls back to ProjectSettings.application/config/version if left empty.

---

int **xr_features/xr_mode** 🔗

The extended reality (XR) mode for this application.

EditorExportPlatformIOS

Inherits: EditorExportPlatform **<** RefCounted **<** Object

Exporter for iOS.

Tutorials

- Exporting for iOS
- iOS plugins documentation index

Properties

String	application/additional_plist_content
String	application/app_store_team_id
String	application/bundle_identifier
String	application/code_sign_identity_debug
String	application/code_sign_identity_release
bool	application/delete_old_export_files_unconditionally
int	application/export_method_debug
int	application/export_method_release
bool	application/export_project_only
bool	application/generate_simulator_library_if_missing

int	application/icon_interpolation
String	application/min_ios_version
String	application/provisioning_profile_uuid_debug
String	application/provisioning_profile_uuid_release
String	application/short_version
String	application/signature
int	application/targeted_device_family
String	application/version
bool	architectures/arm64
bool	capabilities/access_wifi
bool	capabilities/performance_a12
bool	capabilities/performance_gaming_tier
bool	capabilities/push_notifications
String	custom_template/debug
String	custom_template/release
String	icons/app_store_1024x1024

String	icons/ipad_76x76
String	icons/ipad_152x152
String	icons/ipad_167x167
String	icons/iphone_120x120
String	icons/iphone_180x180
String	icons/notification_40x40
String	icons/notification_60x60
String	icons/settings_58x58
String	icons/settings_87x87
String	icons/spotlight_40x40
String	icons/spotlight_80x80
int	privacy/active_keyboard_access_reasons
String	privacy/camera_usage_description
Dictionary	privacy/camera_usage_description_localized
bool	privacy/collected_data/advertising_data/collected
int	privacy/collected_data/advertising_data/collection_purposes

bool	privacy/collected_data/advertising_data/linked_to_user
bool	privacy/collected_data/advertising_data/used_for_tracking
bool	privacy/collected_data/audio_data/collected
int	privacy/collected_data/audio_data/collection_purposes
bool	privacy/collected_data/audio_data/linked_to_user
bool	privacy/collected_data/audio_data/used_for_tracking
bool	privacy/collected_data/browsing_history/collected
int	privacy/collected_data/browsing_history/collection_purposes
bool	privacy/collected_data/browsing_history/linked_to_user
bool	privacy/collected_data/browsing_history/used_for_tracking
bool	privacy/collected_data/coarse_location/collected
int	privacy/collected_data/coarse_location/collection_purposes
bool	privacy/collected_data/coarse_location/linked_to_user
bool	privacy/collected_data/coarse_location/used_for_tracking
bool	privacy/collected_data/contacts/collected
int	privacy/collected_data/contacts/collection_purposes

bool	privacy/collected_data/contacts/linked_to_user
bool	privacy/collected_data/contacts/used_for_tracking
bool	privacy/collected_data/crash_data/collected
int	privacy/collected_data/crash_data/collection_purposes
bool	privacy/collected_data/crash_data/linked_to_user
bool	privacy/collected_data/crash_data/used_for_tracking
bool	privacy/collected_data/credit_info/collected
int	privacy/collected_data/credit_info/collection_purposes
bool	privacy/collected_data/credit_info/linked_to_user
bool	privacy/collected_data/credit_info/used_for_tracking
bool	privacy/collected_data/customer_support/collected
int	privacy/collected_data/customer_support/collection_purposes
bool	privacy/collected_data/customer_support/linked_to_user
bool	privacy/collected_data/customer_support/used_for_tracking
bool	privacy/collected_data/device_id/collected
int	privacy/collected_data/device_id/collection_purposes

bool	privacy/collected_data/device_id/linked_to_user
bool	privacy/collected_data/device_id/used_for_tracking
bool	privacy/collected_data/email_address/collected
int	privacy/collected_data/email_address/collection_purposes
bool	privacy/collected_data/email_address/linked_to_user
bool	privacy/collected_data/email_address/used_for_tracking
bool	privacy/collected_data/emails_or_text_messages/collected
int	privacy/collected_data/emails_or_text_messages/collection_purposes
bool	privacy/collected_data/emails_or_text_messages/linked_to_user
bool	privacy/collected_data/emails_or_text_messages/used_for_tracking
bool	privacy/collected_data/environment_scanning/collected
int	privacy/collected_data/environment_scanning/collection_purposes
bool	privacy/collected_data/environment_scanning/linked_to_user
bool	privacy/collected_data/environment_scanning/used_for_tracking
bool	privacy/collected_data/fitness/collected
int	privacy/collected_data/fitness/collection_purposes

bool	privacy/collected_data/fitness/linked_to_user
bool	privacy/collected_data/fitness/used_for_tracking
bool	privacy/collected_data/gameplay_content/collected
int	privacy/collected_data/gameplay_content/collection_purposes
bool	privacy/collected_data/gameplay_content/linked_to_user
bool	privacy/collected_data/gameplay_content/used_for_tracking
bool	privacy/collected_data/hands/collected
int	privacy/collected_data/hands/collection_purposes
bool	privacy/collected_data/hands/linked_to_user
bool	privacy/collected_data/hands/used_for_tracking
bool	privacy/collected_data/head/collected
int	privacy/collected_data/head/collection_purposes
bool	privacy/collected_data/head/linked_to_user
bool	privacy/collected_data/head/used_for_tracking
bool	privacy/collected_data/health/collected
int	privacy/collected_data/health/collection_purposes

bool	privacy/collected_data/health/linked_to_user
bool	privacy/collected_data/health/used_for_tracking
bool	privacy/collected_data/name/collected
int	privacy/collected_data/name/collection_purposes
bool	privacy/collected_data/name/linked_to_user
bool	privacy/collected_data/name/used_for_tracking
bool	privacy/collected_data/other_contact_info/collected
int	privacy/collected_data/other_contact_info/collection_purposes
bool	privacy/collected_data/other_contact_info/linked_to_user
bool	privacy/collected_data/other_contact_info/used_for_tracking
bool	privacy/collected_data/other_data_types/collected
int	privacy/collected_data/other_data_types/collection_purposes
bool	privacy/collected_data/other_data_types/linked_to_user
bool	privacy/collected_data/other_data_types/used_for_tracking
bool	privacy/collected_data/other_diagnostic_data/collected
int	privacy/collected_data/other_diagnostic_data/collection_purposes

bool	privacy/collected_data/other_diagnostic_data/linked_to_user
bool	privacy/collected_data/other_diagnostic_data/used_for_tracking
bool	privacy/collected_data/other_financial_info/collected
int	privacy/collected_data/other_financial_info/collection_purposes
bool	privacy/collected_data/other_financial_info/linked_to_user
bool	privacy/collected_data/other_financial_info/used_for_tracking
bool	privacy/collected_data/other_usage_data/collected
int	privacy/collected_data/other_usage_data/collection_purposes
bool	privacy/collected_data/other_usage_data/linked_to_user
bool	privacy/collected_data/other_usage_data/used_for_tracking
bool	privacy/collected_data/other_user_content/collected
int	privacy/collected_data/other_user_content/collection_purposes
bool	privacy/collected_data/other_user_content/linked_to_user
bool	privacy/collected_data/other_user_content/used_for_tracking
bool	privacy/collected_data/payment_info/collected
int	privacy/collected_data/payment_info/collection_purposes

bool	privacy/collected_data/payment_info/linked_to_user
bool	privacy/collected_data/payment_info/used_for_tracking
bool	privacy/collected_data/performance_data/collected
int	privacy/collected_data/performance_data/collection_purposes
bool	privacy/collected_data/performance_data/linked_to_user
bool	privacy/collected_data/performance_data/used_for_tracking
bool	privacy/collected_data/phone_number/collected
int	privacy/collected_data/phone_number/collection_purposes
bool	privacy/collected_data/phone_number/linked_to_user
bool	privacy/collected_data/phone_number/used_for_tracking
bool	privacy/collected_data/photos_or_videos/collected
int	privacy/collected_data/photos_or_videos/collection_purposes
bool	privacy/collected_data/photos_or_videos/linked_to_user
bool	privacy/collected_data/photos_or_videos/used_for_tracking
bool	privacy/collected_data/physical_address/collected
int	privacy/collected_data/physical_address/collection_purposes

bool	privacy/collected_data/physical_address/linked_to_user
bool	privacy/collected_data/physical_address/used_for_tracking
bool	privacy/collected_data/precise_location/collected
int	privacy/collected_data/precise_location/collection_purposes
bool	privacy/collected_data/precise_location/linked_to_user
bool	privacy/collected_data/precise_location/used_for_tracking
bool	privacy/collected_data/product_interaction/collected
int	privacy/collected_data/product_interaction/collection_purposes
bool	privacy/collected_data/product_interaction/linked_to_user
bool	privacy/collected_data/product_interaction/used_for_tracking
bool	privacy/collected_data/purchase_history/collected
int	privacy/collected_data/purchase_history/collection_purposes
bool	privacy/collected_data/purchase_history/linked_to_user
bool	privacy/collected_data/purchase_history/used_for_tracking
bool	privacy/collected_data/search_hhistory/collected
int	privacy/collected_data/search_hhistory/collection_purposes

bool	privacy/collected_data/search_hhistory/linked_to_user
bool	privacy/collected_data/search_hhistory/used_for_tracking
bool	privacy/collected_data/sensitive_info/collected
int	privacy/collected_data/sensitive_info/collection_purposes
bool	privacy/collected_data/sensitive_info/linked_to_user
bool	privacy/collected_data/sensitive_info/used_for_tracking
bool	privacy/collected_data/user_id/collected
int	privacy/collected_data/user_id/collection_purposes
bool	privacy/collected_data/user_id/linked_to_user
bool	privacy/collected_data/user_id/used_for_tracking
int	privacy/disk_space_access_reasons
int	privacy/file_timestamp_access_reasons
String	privacy/microphone_usage_description
Dictionary	privacy/microphone_usage_description_localized
String	privacy/photolibrary_usage_description
Dictionary	privacy/photolibrary_usage_description_localized

int	privacy/system_boot_time_access_reasons
PackedStringArray	privacy/tracking_domains
bool	privacy/tracking_enabled
int	privacy/user_defaults_access_reasons
Color	storyboard/custom_bg_color
String	storyboard/custom_image@2x
String	storyboard/custom_image@3x
int	storyboard/image_scale_mode
bool	storyboard/use_custom_bg_color
bool	user_data/accessible_from_files_app
bool	user_data/accessible_from_itunes_sharing

Property Descriptions

String **application/additional_plist_content**

Additional data added to the root `<dict>` section of the Info.plist [https://developer.apple.com/documentation/bundleresources/information_property_list] file. The value should be an XML section with pairs of key-value elements, e.g.:

```
<key>key_name</key>
<string>value</string>
```

String **application/app_store_team_id**

Apple Team ID, unique 10-character string. To locate your Team ID check "Membership details" section in your Apple developer account dashboard, or "Organizational Unit" of your code signing certificate. See Locate your Team ID [https://developer.apple.com/help/account/manage-your-team/locate-your-team-id].

String **application/bundle_identifier** 🔗

Unique application identifier in a reverse-DNS format, can only contain alphanumeric characters (`A-Z`, `a-z`, and `0-9`), hyphens (`-`), and periods (`.`).

String **application/code_sign_identity_debug** 🔗

The "Full Name", "Common Name" or SHA-1 hash of the signing identity used for debug export.

String **application/code_sign_identity_release** 🔗

The "Full Name", "Common Name" or SHA-1 hash of the signing identity used for release export.

---

bool **application/delete_old_export_files_unconditionally** 🔗

If `true`, existing "project name" and "project name.xcodeproj" in the export destination directory will be unconditionally deleted during export.

---

int **application/export_method_debug** 🔗

Application distribution target (debug export).

---

int **application/export_method_release** 🔗

Application distribution target (release export).

---

bool **application/export_project_only** 🔗

If `true`, exports iOS project files without building an XCArchive or `.ipa` file. If `false`, exports iOS project files and builds an XCArchive and `.ipa` file at the same time. When combining Godot with Fastlane or other build pipelines, you may want to set this to `true`.

---

bool **application/generate_simulator_library_if_missing** 🔗

If `true`, and ARM64 simulator library is missing from the export template, it is automatically generated from ARM64 device library.

---

int **application/icon_interpolation** 🔗

Interpolation method used to resize application icon.

String **application/min_ios_version** 🔗

Minimum version of iOS required for this application to run in the `major.minor.patch` or `major.minor` format, can only contain numeric characters (`0-9`) and periods (`.`).

String **application/provisioning_profile_uuid_debug** 🔗

UUID of the provisioning profile. If left empty, Xcode will download or create a provisioning profile automatically. See Edit, download, or delete provisioning profiles [https://developer.apple.com/help/account/manage-profiles/edit-download-or-delete-profiles].

Can be overridden with the environment variable `GODOT_IOS_PROVISIONING_PROFILE_UUID_DEBUG`.

String **application/provisioning_profile_uuid_release** 🔗

UUID of the provisioning profile. If left empty, Xcode will download or create a provisioning profile automatically. See Edit, download, or delete provisioning profiles [https://developer.apple.com/help/account/manage-profiles/edit-download-or-delete-profiles].

Can be overridden with the environment variable `GODOT_IOS_PROVISIONING_PROFILE_UUID_RELEASE`.

String **application/short_version** 🔗

Application version visible to the user, can only contain numeric characters (`0-9`) and periods (`.`). Falls back to ProjectSettings.application/config/version if left empty.

String **application/signature** 🔗

A four-character creator code that is specific to the bundle. Optional.

---

int **application/targeted_device_family** 🔗

Supported device family.

String **application/version** 🔗

Machine-readable application version, in the `major.minor.patch` format, can only contain numeric characters (`0-9`) and periods (`.`). This must be incremented on every new release pushed to the App Store.

---

bool **architectures/arm64** 🔗

If `true`, `arm64` binaries are included into exported project.

---

bool **capabilities/access_wifi** 🔗

If `true`, networking features related to Wi-Fi access are enabled. See Required Device Capabilities [https://developer.apple.com/support/required-device-capabilities/].

---

bool **capabilities/performance_a12** 🔗

Requires the graphics performance and features of the A12 Bionic and later chips (devices supporting all Vulkan renderer features).

Enabling this option limits supported devices to: iPhone XS, iPhone XR, iPad Mini (5th gen.), iPad Air (3rd gen.), iPad (8th gen) and newer.

---

bool **capabilities/performance_gaming_tier** 🔗

Requires the graphics performance and features of the A17 Pro and later chips.

Enabling this option limits supported devices to: iPhone 15 Pro and newer.

---

bool **capabilities/push_notifications** 🔗

If `true`, push notifications are enabled. See Required Device Capabilities [https://developer.apple.com/support/required-device-capabilities/].

String **custom_template/debug** 🔗

Path to the custom export template. If left empty, default template is used.

String **custom_template/release**

Path to the custom export template. If left empty, default template is used.

String **icons/app_store_1024x1024**

App Store application icon file. If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/ipad_76x76**

Home screen application icon file on iPad (1x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/ipad_152x152**

Home screen application icon file on iPad (2x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/ipad_167x167**

Home screen application icon file on iPad (3x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/iphone_120x120**

Home screen application icon file on iPhone (2x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/iphone_180x180**

Home screen application icon file on iPhone (3x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/notification_40x40**

Notification icon file on iPad and iPhone (2x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/notification_60x60**

Notification icon file on iPhone (3x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/settings_58x58**

Application settings icon file on iPad and iPhone (2x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/settings_87x87**

Application settings icon file on iPhone (3x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/spotlight_40x40**

Spotlight icon file on iPad (1x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

String **icons/spotlight_80x80**

Spotlight icon file on iPad and iPhone (2x DPI). If left empty, it will fallback to ProjectSettings.application/config/icon. See App icons [https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons].

---

int **privacy/active_keyboard_access_reasons**

The reasons your app use active keyboard API. See Describing use of required reason API [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_use_of_required_reason_api].

String privacy/camera_usage_description

A message displayed when requesting access to the device's camera (in English).

Dictionary privacy/camera_usage_description_localized

A message displayed when requesting access to the device's camera (localized).

bool privacy/collected_data/advertising_data/collected

Indicates whether your app collects advertising data.

int privacy/collected_data/advertising_data/collection_purposes

The reasons your app collects advertising data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool privacy/collected_data/advertising_data/linked_to_user

Indicates whether your app links advertising data to the user's identity.

bool privacy/collected_data/advertising_data/used_for_tracking

Indicates whether your app uses advertising data for tracking.

bool privacy/collected_data/audio_data/collected

Indicates whether your app collects audio data.

int privacy/collected_data/audio_data/collection_purposes

The reasons your app collects audio data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/audio_data/linked_to_user**

Indicates whether your app links audio data to the user's identity.

---

bool **privacy/collected_data/audio_data/used_for_tracking**

Indicates whether your app uses audio data for tracking.

---

bool **privacy/collected_data/browsing_history/collected**

Indicates whether your app collects browsing history.

---

int **privacy/collected_data/browsing_history/collection_purposes**

The reasons your app collects browsing history. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/browsing_history/linked_to_user**

Indicates whether your app links browsing history to the user's identity.

---

bool **privacy/collected_data/browsing_history/used_for_tracking**

Indicates whether your app uses browsing history for tracking.

---

bool **privacy/collected_data/coarse_location/collected**

Indicates whether your app collects coarse location data.

---

int **privacy/collected_data/coarse_location/collection_purposes**

The reasons your app collects coarse location data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/coarse_location/linked_to_user**

Indicates whether your app links coarse location data to the user's identity.

bool privacy/collected_data/coarse_location/used_for_tracking

Indicates whether your app uses coarse location data for tracking.

bool privacy/collected_data/contacts/collected

Indicates whether your app collects contacts.

int privacy/collected_data/contacts/collection_purposes

The reasons your app collects contacts. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool privacy/collected_data/contacts/linked_to_user

Indicates whether your app links contacts to the user's identity.

bool privacy/collected_data/contacts/used_for_tracking

Indicates whether your app uses contacts for tracking.

bool privacy/collected_data/crash_data/collected

Indicates whether your app collects crash data.

int privacy/collected_data/crash_data/collection_purposes

The reasons your app collects crash data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool privacy/collected_data/crash_data/linked_to_user

Indicates whether your app links crash data to the user's identity.

bool privacy/collected_data/crash_data/used_for_tracking

Indicates whether your app uses crash data for tracking.

---

bool **privacy/collected_data/credit_info/collected** 🔗

Indicates whether your app collects credit information.

---

int **privacy/collected_data/credit_info/collection_purposes** 🔗

The reasons your app collects credit information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/credit_info/linked_to_user** 🔗

Indicates whether your app links credit information to the user's identity.

---

bool **privacy/collected_data/credit_info/used_for_tracking** 🔗

Indicates whether your app uses credit information for tracking.

---

bool **privacy/collected_data/customer_support/collected** 🔗

Indicates whether your app collects customer support data.

---

int **privacy/collected_data/customer_support/collection_purposes** 🔗

The reasons your app collects customer support data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/customer_support/linked_to_user** 🔗

Indicates whether your app links customer support data to the user's identity.

---

bool **privacy/collected_data/customer_support/used_for_tracking** 🔗

Indicates whether your app uses customer support data for tracking.

---

bool **privacy/collected_data/device_id/collected** 🔗

Indicates whether your app collects device IDs.

int **privacy/collected_data/device_id/collection_purposes** 🔗

The reasons your app collects device IDs. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/device_id/linked_to_user** 🔗

Indicates whether your app links device IDs to the user's identity.

bool **privacy/collected_data/device_id/used_for_tracking** 🔗

Indicates whether your app uses device IDs for tracking.

bool **privacy/collected_data/email_address/collected** 🔗

Indicates whether your app collects email address.

int **privacy/collected_data/email_address/collection_purposes** 🔗

The reasons your app collects email address. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/email_address/linked_to_user** 🔗

Indicates whether your app links email address to the user's identity.

bool **privacy/collected_data/email_address/used_for_tracking** 🔗

Indicates whether your app uses email address for tracking.

bool **privacy/collected_data/emails_or_text_messages/collected** 🔗

Indicates whether your app collects emails or text messages.

int privacy/collected_data/emails_or_text_messages/collection_purposes

The reasons your app collects emails or text messages. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool privacy/collected_data/emails_or_text_messages/linked_to_user

Indicates whether your app links emails or text messages to the user's identity.

bool privacy/collected_data/emails_or_text_messages/used_for_tracking

Indicates whether your app uses emails or text messages for tracking.

bool privacy/collected_data/environment_scanning/collected

Indicates whether your app collects environment scanning data.

int privacy/collected_data/environment_scanning/collection_purposes

The reasons your app collects environment scanning data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool privacy/collected_data/environment_scanning/linked_to_user

Indicates whether your app links environment scanning data to the user's identity.

bool privacy/collected_data/environment_scanning/used_for_tracking

Indicates whether your app uses environment scanning data for tracking.

bool privacy/collected_data/fitness/collected

Indicates whether your app collects fitness and exercise data.

int **privacy/collected_data/fitness/collection_purposes** 🔗

The reasons your app collects fitness and exercise data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/fitness/linked_to_user** 🔗

Indicates whether your app links fitness and exercise data to the user's identity.

---

bool **privacy/collected_data/fitness/used_for_tracking** 🔗

Indicates whether your app uses fitness and exercise data for tracking.

---

bool **privacy/collected_data/gameplay_content/collected** 🔗

Indicates whether your app collects gameplay content.

---

int **privacy/collected_data/gameplay_content/collection_purposes** 🔗

The reasons your app collects gameplay content. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/gameplay_content/linked_to_user** 🔗

Indicates whether your app links gameplay content to the user's identity.

---

bool **privacy/collected_data/gameplay_content/used_for_tracking** 🔗

Indicates whether your app uses gameplay content for tracking.

---

bool **privacy/collected_data/hands/collected** 🔗

Indicates whether your app collects user's hand structure and hand movements.

---

int **privacy/collected_data/hands/collection_purposes** 🔗

The reasons your app collects user's hand structure and hand movements. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/hands/linked_to_user**

Indicates whether your app links user's hand structure and hand movements to the user's identity.

bool **privacy/collected_data/hands/used_for_tracking**

Indicates whether your app uses user's hand structure and hand movements for tracking.

bool **privacy/collected_data/head/collected**

Indicates whether your app collects user's head movement.

int **privacy/collected_data/head/collection_purposes**

The reasons your app collects user's head movement. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/head/linked_to_user**

Indicates whether your app links user's head movement to the user's identity.

bool **privacy/collected_data/head/used_for_tracking**

Indicates whether your app uses user's head movement for tracking.

bool **privacy/collected_data/health/collected**

Indicates whether your app collects health and medical data.

int **privacy/collected_data/health/collection_purposes**

The reasons your app collects health and medical data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/health/linked_to_user** 🔗

Indicates whether your app links health and medical data to the user's identity.

---

bool **privacy/collected_data/health/used_for_tracking** 🔗

Indicates whether your app uses health and medical data for tracking.

---

bool **privacy/collected_data/name/collected** 🔗

Indicates whether your app collects user's name.

---

int **privacy/collected_data/name/collection_purposes** 🔗

The reasons your app collects user's name. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/name/linked_to_user** 🔗

Indicates whether your app links user's name to the user's identity.

---

bool **privacy/collected_data/name/used_for_tracking** 🔗

Indicates whether your app uses user's name for tracking.

---

bool **privacy/collected_data/other_contact_info/collected** 🔗

Indicates whether your app collects any other contact information.

---

int **privacy/collected_data/other_contact_info/collection_purposes** 🔗

The reasons your app collects any other contact information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/other_contact_info/linked_to_user** 🔗

Indicates whether your app links any other contact information to the user's identity.

---

bool **privacy/collected_data/other_contact_info/used_for_tracking** 🔗

Indicates whether your app uses any other contact information for tracking.

---

bool **privacy/collected_data/other_data_types/collected** 🔗

Indicates whether your app collects any other data.

---

int **privacy/collected_data/other_data_types/collection_purposes** 🔗

The reasons your app collects any other data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/other_data_types/linked_to_user** 🔗

Indicates whether your app links any other data to the user's identity.

---

bool **privacy/collected_data/other_data_types/used_for_tracking** 🔗

Indicates whether your app uses any other data for tracking.

---

bool **privacy/collected_data/other_diagnostic_data/collected** 🔗

Indicates whether your app collects any other diagnostic data.

---

int **privacy/collected_data/other_diagnostic_data/collection_purposes** 🔗

The reasons your app collects any other diagnostic data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/other_diagnostic_data/linked_to_user**

Indicates whether your app links any other diagnostic data to the user's identity.

bool **privacy/collected_data/other_diagnostic_data/used_for_tracking**

Indicates whether your app uses any other diagnostic data for tracking.

bool **privacy/collected_data/other_financial_info/collected**

Indicates whether your app collects any other financial information.

int **privacy/collected_data/other_financial_info/collection_purposes**

The reasons your app collects any other financial information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/other_financial_info/linked_to_user**

Indicates whether your app links any other financial information to the user's identity.

bool **privacy/collected_data/other_financial_info/used_for_tracking**

Indicates whether your app uses any other financial information for tracking.

bool **privacy/collected_data/other_usage_data/collected**

Indicates whether your app collects any other usage data.

int **privacy/collected_data/other_usage_data/collection_purposes**

The reasons your app collects any other usage data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/other_usage_data/linked_to_user** 🔗

Indicates whether your app links any other usage data to the user's identity.

bool **privacy/collected_data/other_usage_data/used_for_tracking** 🔗

Indicates whether your app uses any other usage data for tracking.

bool **privacy/collected_data/other_user_content/collected** 🔗

Indicates whether your app collects any other user generated content.

int **privacy/collected_data/other_user_content/collection_purposes** 🔗

The reasons your app collects any other user generated content. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/other_user_content/linked_to_user** 🔗

Indicates whether your app links any other user generated content to the user's identity.

bool **privacy/collected_data/other_user_content/used_for_tracking** 🔗

Indicates whether your app uses any other user generated content for tracking.

bool **privacy/collected_data/payment_info/collected** 🔗

Indicates whether your app collects payment information.

int **privacy/collected_data/payment_info/collection_purposes** 🔗

The reasons your app collects payment information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/payment_info/linked_to_user** 🔗

Indicates whether your app links payment information to the user's identity.

---

bool **privacy/collected_data/payment_info/used_for_tracking** 🔗

Indicates whether your app uses payment information for tracking.

---

bool **privacy/collected_data/performance_data/collected** 🔗

Indicates whether your app collects performance data.

---

int **privacy/collected_data/performance_data/collection_purposes** 🔗

The reasons your app collects performance data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/performance_data/linked_to_user** 🔗

Indicates whether your app links performance data to the user's identity.

---

bool **privacy/collected_data/performance_data/used_for_tracking** 🔗

Indicates whether your app uses performance data for tracking.

---

bool **privacy/collected_data/phone_number/collected** 🔗

Indicates whether your app collects phone number.

---

int **privacy/collected_data/phone_number/collection_purposes** 🔗

The reasons your app collects phone number. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/phone_number/linked_to_user** 🔗

Indicates whether your app links phone number to the user's identity.

---

bool **privacy/collected_data/phone_number/used_for_tracking**

Indicates whether your app uses phone number for tracking.

---

bool **privacy/collected_data/photos_or_videos/collected**

Indicates whether your app collects photos or videos.

---

int **privacy/collected_data/photos_or_videos/collection_purposes**

The reasons your app collects photos or videos. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/photos_or_videos/linked_to_user**

Indicates whether your app links photos or videos to the user's identity.

---

bool **privacy/collected_data/photos_or_videos/used_for_tracking**

Indicates whether your app uses photos or videos for tracking.

---

bool **privacy/collected_data/physical_address/collected**

Indicates whether your app collects physical address.

---

int **privacy/collected_data/physical_address/collection_purposes**

The reasons your app collects physical address. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/physical_address/linked_to_user**

Indicates whether your app links physical address to the user's identity.

---

bool **privacy/collected_data/physical_address/used_for_tracking**

Indicates whether your app uses physical address for tracking.

bool **privacy/collected_data/precise_location/collected** 🔗

Indicates whether your app collects precise location data.

int **privacy/collected_data/precise_location/collection_purposes** 🔗

The reasons your app collects precise location data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/precise_location/linked_to_user** 🔗

Indicates whether your app links precise location data to the user's identity.

bool **privacy/collected_data/precise_location/used_for_tracking** 🔗

Indicates whether your app uses precise location data for tracking.

bool **privacy/collected_data/product_interaction/collected** 🔗

Indicates whether your app collects product interaction data.

int **privacy/collected_data/product_interaction/collection_purposes** 🔗

The reasons your app collects product interaction data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/product_interaction/linked_to_user** 🔗

Indicates whether your app links product interaction data to the user's identity.

bool **privacy/collected_data/product_interaction/used_for_tracking** 🔗

Indicates whether your app uses product interaction data for tracking.

bool **privacy/collected_data/purchase_history/collected**

Indicates whether your app collects purchase history.

int **privacy/collected_data/purchase_history/collection_purposes**

The reasons your app collects purchase history. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/purchase_history/linked_to_user**

Indicates whether your app links purchase history to the user's identity.

bool **privacy/collected_data/purchase_history/used_for_tracking**

Indicates whether your app uses purchase history for tracking.

bool **privacy/collected_data/search_hhistory/collected**

Indicates whether your app collects search history.

int **privacy/collected_data/search_hhistory/collection_purposes**

The reasons your app collects search history. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/search_hhistory/linked_to_user**

Indicates whether your app links search history to the user's identity.

bool **privacy/collected_data/search_hhistory/used_for_tracking**

Indicates whether your app uses search history for tracking.

bool **privacy/collected_data/sensitive_info/collected**

Indicates whether your app collects sensitive user information.

int **privacy/collected_data/sensitive_info/collection_purposes**

The reasons your app collects sensitive user information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/sensitive_info/linked_to_user**

Indicates whether your app links sensitive user information to the user's identity.

bool **privacy/collected_data/sensitive_info/used_for_tracking**

Indicates whether your app uses sensitive user information for tracking.

bool **privacy/collected_data/user_id/collected**

Indicates whether your app collects user IDs.

int **privacy/collected_data/user_id/collection_purposes**

The reasons your app collects user IDs. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/user_id/linked_to_user**

Indicates whether your app links user IDs to the user's identity.

bool **privacy/collected_data/user_id/used_for_tracking**

Indicates whether your app uses user IDs for tracking.

int **privacy/disk_space_access_reasons**

The reasons your app use free disk space API. See Describing use of required reason API [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_use_of_required_reason_api].

---

int **privacy/file_timestamp_access_reasons** 🔗

The reasons your app use file timestamp/metadata API. See Describing use of required reason API [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_use_of_required_reason_api].

String **privacy/microphone_usage_description** 🔗

A message displayed when requesting access to the device's microphone (in English).

Dictionary **privacy/microphone_usage_description_localized** 🔗

A message displayed when requesting access to the device's microphone (localized).

String **privacy/photolibrary_usage_description** 🔗

A message displayed when requesting access to the user's photo library (in English).

Dictionary **privacy/photolibrary_usage_description_localized** 🔗

A message displayed when requesting access to the user's photo library (localized).

---

int **privacy/system_boot_time_access_reasons** 🔗

The reasons your app use system boot time / absolute time API. See Describing use of required reason API [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_use_of_required_reason_api].

PackedStringArray **privacy/tracking_domains** 🔗

The list of internet domains your app connects to that engage in tracking. See Privacy manifest files [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files].

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedStringArray for more details.

---

bool **privacy/tracking_enabled** 🔗

Indicates whether your app uses data for tracking. See Privacy manifest files [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files].

---

int **privacy/user_defaults_access_reasons**

The reasons your app use user defaults API. See Describing use of required reason API [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_use_of_required_reason_api].

Color **storyboard/custom_bg_color**

A custom background color of the storyboard launch screen.

String **storyboard/custom_image@2x**

Application launch screen image file (2x DPI). If left empty, it will fallback to ProjectSettings.application/boot_splash/image.

String **storyboard/custom_image@3x**

Application launch screen image file (3x DPI). If left empty, it will fallback to ProjectSettings.application/boot_splash/image.

---

int **storyboard/image_scale_mode**

Launch screen image scaling mode.

---

bool **storyboard/use_custom_bg_color**

If `true`, storyboard/custom_bg_color is used as a launch screen background color, otherwise `application/boot_splash/bg_color` project setting is used.

---

bool **user_data/accessible_from_files_app**

If `true`, the app "Documents" folder can be accessed via "Files" app. See LSSupportsOpeningDocumentsInPlace [https://developer.apple.com/documentation/bundleresources/information_property_list/lssupportsopeningdocumentsinplace].

---

bool **user_data/accessible_from_itunes_sharing** 🔗

If `true`, the app "Documents" folder can be accessed via iTunes file sharing. See UIFileSharingEnabled [https://developer.apple.com/documentation/bundleresources/information_property_list/uifilesharingenabled].

EditorExportPlatformLinuxBSD

Inherits: EditorExportPlatformPC **<** EditorExportPlatform **<** RefCounted **<** Object

Exporter for Linux/BSD.

Tutorials

- Exporting for Linux

Properties

String	binary_format/architecture
bool	binary_format/embed_pck
String	custom_template/debug
String	custom_template/release
int	debug/export_console_wrapper
String	ssh_remote_deploy/cleanup_script

bool	ssh_remote_deploy/enabled
String	ssh_remote_deploy/extra_args_scp
String	ssh_remote_deploy/extra_args_ssh
String	ssh_remote_deploy/host
String	ssh_remote_deploy/port
String	ssh_remote_deploy/run_script
bool	texture_format/etc2_astc
bool	texture_format/s3tc_bptc

Property Descriptions

String **binary_format/architecture**

Application executable architecture.

Supported architectures: `x86_32`, `x86_64`, `arm64`, `arm32`, `rv64`, `ppc64`, and `ppc32`.

Official export templates include `x86_32` and `x86_64` binaries only.

bool **binary_format/embed_pck** 🔗

If `true`, project resources are embedded into the executable.

String **custom_template/debug** 🔗

Path to the custom export template. If left empty, default template is used.

String **custom_template/release** 🔗

Path to the custom export template. If left empty, default template is used.

---

int **debug/export_console_wrapper** 🔗

If `true`, a console wrapper is exported alongside the main executable, which allows running the project with enabled console output.

String **ssh_remote_deploy/cleanup_script** 🔗

Script code to execute on the remote host when app is finished.

The following variables can be used in the script:

- `{temp_dir}` - Path of temporary folder on the remote, used to upload app and scripts to.
- `{archive_name}` - Name of the ZIP containing uploaded application.
- `{exe_name}` - Name of application executable.

- `{cmd_args}` - Array of the command line argument for the application.

---

bool **ssh_remote_deploy/enabled** 🔗

Enables remote deploy using SSH/SCP.

String **ssh_remote_deploy/extra_args_scp** 🔗

Array of the additional command line arguments passed to the SCP.

String **ssh_remote_deploy/extra_args_ssh** 🔗

Array of the additional command line arguments passed to the SSH.

String **ssh_remote_deploy/host** 🔗

Remote host SSH user name and address, in `user@address` format.

String **ssh_remote_deploy/port** 🔗

Remote host SSH port number.

String **ssh_remote_deploy/run_script** 🔗

Script code to execute on the remote host when running the app.

The following variables can be used in the script:

- `{temp_dir}` - Path of temporary folder on the remote, used to upload app and scripts to.

- `{archive_name}` - Name of the ZIP containing uploaded application.
- `{exe_name}` - Name of application executable.
- `{cmd_args}` - Array of the command line argument for the application.

---

bool **texture_format/etc2_astc**

If `true`, project textures are exported in the ETC2/ASTC format.

---

bool **texture_format/s3tc_bptc**

If `true`, project textures are exported in the S3TC/BPTC format.

EditorExportPlatformMacOS

Inherits: EditorExportPlatform **<** RefCounted **<** Object

Exporter for macOS.

Tutorials

- Exporting for macOS
- Running Godot apps on macOS

Properties

String	application/additional_plist_content
String	application/app_category
String	application/bundle_identifier
String	application/copyright
Dictionary	application/copyright_localized
int	application/export_angle
String	application/icon
int	application/icon_interpolation
String	application/min_macos_version
String	application/short_version

String	application/signature
String	application/version
String	binary_format/architecture
String	codesign/apple_team_id
String	codesign/certificate_file
String	codesign/certificate_password
int	codesign/codesign
PackedStringArray	codesign/custom_options
bool	codesign/entitlements/address_book
bool	codesign/entitlements/allow_dyld_environment_variables
bool	codesign/entitlements/allow_jit_code_execution
bool	codesign/entitlements/allow_unsigned_executable_memory
bool	codesign/entitlements/app_sandbox/device_bluetooth
bool	codesign/entitlements/app_sandbox/device_usb
bool	codesign/entitlements/app_sandbox/enabled
int	codesign/entitlements/app_sandbox/files_downloads

int	codesign/entitlements/app_sandbox/files_movies
int	codesign/entitlements/app_sandbox/files_music
int	codesign/entitlements/app_sandbox/files_pictures
int	codesign/entitlements/app_sandbox/files_user_selected
Array	codesign/entitlements/app_sandbox/helper_executables
bool	codesign/entitlements/app_sandbox/network_client
bool	codesign/entitlements/app_sandbox/network_server
bool	codesign/entitlements/apple_events
bool	codesign/entitlements/audio_input
bool	codesign/entitlements/calendars
bool	codesign/entitlements/camera
String	codesign/entitlements/custom_file
bool	codesign/entitlements/debugging
bool	codesign/entitlements/disable_library_validation
bool	codesign/entitlements/location
bool	codesign/entitlements/photos_library

String	codesign/identity
String	codesign/installer_identity
String	codesign/provisioning_profile
String	custom_template/debug
String	custom_template/release
int	debug/export_console_wrapper
bool	display/high_res
int	export/distribution_type
String	notarization/api_key
String	notarization/api_key_id
String	notarization/api_uuid
String	notarization/apple_id_name
String	notarization/apple_id_password
int	notarization/notarization
String	privacy/address_book_usage_description
Dictionary	privacy/address_book_usage_description_localized

String	privacy/calendar_usage_description
Dictionary	privacy/calendar_usage_description_localized
String	privacy/camera_usage_description
Dictionary	privacy/camera_usage_description_localized
bool	privacy/collected_data/advertising_data/collected
int	privacy/collected_data/advertising_data/collection_purposes
bool	privacy/collected_data/advertising_data/linked_to_user
bool	privacy/collected_data/advertising_data/used_for_tracking
bool	privacy/collected_data/audio_data/collected
int	privacy/collected_data/audio_data/collection_purposes
bool	privacy/collected_data/audio_data/linked_to_user
bool	privacy/collected_data/audio_data/used_for_tracking
bool	privacy/collected_data/browsing_history/collected
int	privacy/collected_data/browsing_history/collection_purposes
bool	privacy/collected_data/browsing_history/linked_to_user
bool	privacy/collected_data/browsing_history/used_for_tracking

bool	privacy/collected_data/coarse_location/collected
int	privacy/collected_data/coarse_location/collection_purposes
bool	privacy/collected_data/coarse_location/linked_to_user
bool	privacy/collected_data/coarse_location/used_for_tracking
bool	privacy/collected_data/contacts/collected
int	privacy/collected_data/contacts/collection_purposes
bool	privacy/collected_data/contacts/linked_to_user
bool	privacy/collected_data/contacts/used_for_tracking
bool	privacy/collected_data/crash_data/collected
int	privacy/collected_data/crash_data/collection_purposes
bool	privacy/collected_data/crash_data/linked_to_user
bool	privacy/collected_data/crash_data/used_for_tracking
bool	privacy/collected_data/credit_info/collected
int	privacy/collected_data/credit_info/collection_purposes
bool	privacy/collected_data/credit_info/linked_to_user
bool	privacy/collected_data/credit_info/used_for_tracking

bool	privacy/collected_data/customer_support/collected
int	privacy/collected_data/customer_support/collection_purposes
bool	privacy/collected_data/customer_support/linked_to_user
bool	privacy/collected_data/customer_support/used_for_tracking
bool	privacy/collected_data/device_id/collected
int	privacy/collected_data/device_id/collection_purposes
bool	privacy/collected_data/device_id/linked_to_user
bool	privacy/collected_data/device_id/used_for_tracking
bool	privacy/collected_data/email_address/collected
int	privacy/collected_data/email_address/collection_purposes
bool	privacy/collected_data/email_address/linked_to_user
bool	privacy/collected_data/email_address/used_for_tracking
bool	privacy/collected_data/emails_or_text_messages/collected
int	privacy/collected_data/emails_or_text_messages/collection_purposes
bool	privacy/collected_data/emails_or_text_messages/linked_to_user
bool	privacy/collected_data/emails_or_text_messages/used_for_tracking

bool	privacy/collected_data/environment_scanning/collected
int	privacy/collected_data/environment_scanning/collection_purposes
bool	privacy/collected_data/environment_scanning/linked_to_user
bool	privacy/collected_data/environment_scanning/used_for_tracking
bool	privacy/collected_data/fitness/collected
int	privacy/collected_data/fitness/collection_purposes
bool	privacy/collected_data/fitness/linked_to_user
bool	privacy/collected_data/fitness/used_for_tracking
bool	privacy/collected_data/gameplay_content/collected
int	privacy/collected_data/gameplay_content/collection_purposes
bool	privacy/collected_data/gameplay_content/linked_to_user
bool	privacy/collected_data/gameplay_content/used_for_tracking
bool	privacy/collected_data/hands/collected
int	privacy/collected_data/hands/collection_purposes
bool	privacy/collected_data/hands/linked_to_user
bool	privacy/collected_data/hands/used_for_tracking

bool	privacy/collected_data/head/collected
int	privacy/collected_data/head/collection_purposes
bool	privacy/collected_data/head/linked_to_user
bool	privacy/collected_data/head/used_for_tracking
bool	privacy/collected_data/health/collected
int	privacy/collected_data/health/collection_purposes
bool	privacy/collected_data/health/linked_to_user
bool	privacy/collected_data/health/used_for_tracking
bool	privacy/collected_data/name/collected
int	privacy/collected_data/name/collection_purposes
bool	privacy/collected_data/name/linked_to_user
bool	privacy/collected_data/name/used_for_tracking
bool	privacy/collected_data/other_contact_info/collected
int	privacy/collected_data/other_contact_info/collection_purposes
bool	privacy/collected_data/other_contact_info/linked_to_user
bool	privacy/collected_data/other_contact_info/used_for_tracking

bool	privacy/collected_data/other_data_types/collected
int	privacy/collected_data/other_data_types/collection_purposes
bool	privacy/collected_data/other_data_types/linked_to_user
bool	privacy/collected_data/other_data_types/used_for_tracking
bool	privacy/collected_data/other_diagnostic_data/collected
int	privacy/collected_data/other_diagnostic_data/collection_purposes
bool	privacy/collected_data/other_diagnostic_data/linked_to_user
bool	privacy/collected_data/other_diagnostic_data/used_for_tracking
bool	privacy/collected_data/other_financial_info/collected
int	privacy/collected_data/other_financial_info/collection_purposes
bool	privacy/collected_data/other_financial_info/linked_to_user
bool	privacy/collected_data/other_financial_info/used_for_tracking
bool	privacy/collected_data/other_usage_data/collected
int	privacy/collected_data/other_usage_data/collection_purposes
bool	privacy/collected_data/other_usage_data/linked_to_user
bool	privacy/collected_data/other_usage_data/used_for_tracking

bool	privacy/collected_data/other_user_content/collected
int	privacy/collected_data/other_user_content/collection_purposes
bool	privacy/collected_data/other_user_content/linked_to_user
bool	privacy/collected_data/other_user_content/used_for_tracking
bool	privacy/collected_data/payment_info/collected
int	privacy/collected_data/payment_info/collection_purposes
bool	privacy/collected_data/payment_info/linked_to_user
bool	privacy/collected_data/payment_info/used_for_tracking
bool	privacy/collected_data/performance_data/collected
int	privacy/collected_data/performance_data/collection_purposes
bool	privacy/collected_data/performance_data/linked_to_user
bool	privacy/collected_data/performance_data/used_for_tracking
bool	privacy/collected_data/phone_number/collected
int	privacy/collected_data/phone_number/collection_purposes
bool	privacy/collected_data/phone_number/linked_to_user
bool	privacy/collected_data/phone_number/used_for_tracking

bool	privacy/collected_data/photos_or_videos/collected
int	privacy/collected_data/photos_or_videos/collection_purposes
bool	privacy/collected_data/photos_or_videos/linked_to_user
bool	privacy/collected_data/photos_or_videos/used_for_tracking
bool	privacy/collected_data/physical_address/collected
int	privacy/collected_data/physical_address/collection_purposes
bool	privacy/collected_data/physical_address/linked_to_user
bool	privacy/collected_data/physical_address/used_for_tracking
bool	privacy/collected_data/precise_location/collected
int	privacy/collected_data/precise_location/collection_purposes
bool	privacy/collected_data/precise_location/linked_to_user
bool	privacy/collected_data/precise_location/used_for_tracking
bool	privacy/collected_data/product_interaction/collected
int	privacy/collected_data/product_interaction/collection_purposes
bool	privacy/collected_data/product_interaction/linked_to_user
bool	privacy/collected_data/product_interaction/used_for_tracking

bool	privacy/collected_data/purchase_history/collected
int	privacy/collected_data/purchase_history/collection_purposes
bool	privacy/collected_data/purchase_history/linked_to_user
bool	privacy/collected_data/purchase_history/used_for_tracking
bool	privacy/collected_data/search_hhistory/collected
int	privacy/collected_data/search_hhistory/collection_purposes
bool	privacy/collected_data/search_hhistory/linked_to_user
bool	privacy/collected_data/search_hhistory/used_for_tracking
bool	privacy/collected_data/sensitive_info/collected
int	privacy/collected_data/sensitive_info/collection_purposes
bool	privacy/collected_data/sensitive_info/linked_to_user
bool	privacy/collected_data/sensitive_info/used_for_tracking
bool	privacy/collected_data/user_id/collected
int	privacy/collected_data/user_id/collection_purposes
bool	privacy/collected_data/user_id/linked_to_user
bool	privacy/collected_data/user_id/used_for_tracking

String	privacy/desktop_folder_usage_description
Dictionary	privacy/desktop_folder_usage_description_localized
String	privacy/documents_folder_usage_description
Dictionary	privacy/documents_folder_usage_description_localized
String	privacy/downloads_folder_usage_description
Dictionary	privacy/downloads_folder_usage_description_localized
String	privacy/location_usage_description
Dictionary	privacy/location_usage_description_localized
String	privacy/microphone_usage_description
Dictionary	privacy/microphone_usage_description_localized
String	privacy/network_volumes_usage_description
Dictionary	privacy/network_volumes_usage_description_localized
String	privacy/photos_library_usage_description
Dictionary	privacy/photos_library_usage_description_localized
String	privacy/removable_volumes_usage_description
Dictionary	privacy/removable_volumes_usage_description_localized

PackedStringArray	privacy/tracking_domains
bool	privacy/tracking_enabled
String	ssh_remote_deploy/cleanup_script
bool	ssh_remote_deploy/enabled
String	ssh_remote_deploy/extra_args_scp
String	ssh_remote_deploy/extra_args_ssh
String	ssh_remote_deploy/host
String	ssh_remote_deploy/port
String	ssh_remote_deploy/run_script
String	xcode/platform_build
String	xcode/sdk_build
String	xcode/sdk_name
String	xcode/sdk_version
String	xcode/xcode_build
String	xcode/xcode_version

Property Descriptions

String **application/additional_plist_content** 🔗

Additional data added to the root `<dict>` section of the Info.plist [https://developer.apple.com/documentation/bundleresources/information_property_list] file. The value should be an XML section with pairs of key-value elements, e.g.:

```
<key>key_name</key>
<string>value</string>
```

String **application/app_category** 🔗

Application category for the App Store.

String **application/bundle_identifier** 🔗

Unique application identifier in a reverse-DNS format, can only contain alphanumeric characters (`A-Z`, `a-z`, and `0-9`), hyphens (`-`), and periods (`.`).

String **application/copyright** 🔗

Copyright notice for the bundle visible to the user (in English).

Dictionary **application/copyright_localized** 🔗

Copyright notice for the bundle visible to the user (localized).

---

int **application/export_angle** 🔗

If set to `1`, ANGLE libraries are exported with the exported application. If set to `0`, ANGLE libraries are exported only if ProjectSettings.rendering/gl_compatibility/driver is set to `"opengl3_angle"`.

String **application/icon** 🔗

Application icon file. If left empty, it will fallback to ProjectSettings.application/config/macos_native_icon, and then to ProjectSettings.application/config/icon.

---

int **application/icon_interpolation** 🔗

Interpolation method used to resize application icon.

String **application/min_macos_version**

Minimum version of macOS required for this application to run in the `major.minor.patch` or `major.minor` format, can only contain numeric characters (`0-9`) and periods (`.`).

String **application/short_version**

Application version visible to the user, can only contain numeric characters (`0-9`) and periods (`.`). Falls back to ProjectSettings.application/config/version if left empty.

String **application/signature**

A four-character creator code that is specific to the bundle. Optional.

String **application/version**

Machine-readable application version, in the `major.minor.patch` format, can only contain numeric characters (`0-9`) and periods (`.`). This must be incremented on every new release pushed to the App Store.

String **binary_format/architecture**

Application executable architecture.

Supported architectures: `x86_64`, `arm64`, and `universal` (`x86_64 + arm64`).

Official export templates include `universal` binaries only.

String **codesign/apple_team_id**

Apple Team ID, unique 10-character string. To locate your Team ID check "Membership details" section in your Apple developer account dashboard, or "Organizational Unit" of your code signing certificate. See Locate your Team ID [https://developer.apple.com/help/account/manage-your-team/locate-your-team-id].

String **codesign/certificate_file**

PKCS #12 certificate file used to sign `.app` bundle.

Can be overridden with the environment variable `GODOT_MACOS_CODESIGN_CERTIFICATE_FILE`.

String **codesign/certificate_password** 🔗

Password for the certificate file used to sign `.app` bundle.

Can be overridden with the environment variable `GODOT_MACOS_CODESIGN_CERTIFICATE_PASSWORD`.

---

int **codesign/codesign** 🔗

Tool to use for code signing.

PackedStringArray **codesign/custom_options** 🔗

Array of the additional command line arguments passed to the code signing tool.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedStringArray for more details.

---

bool **codesign/entitlements/address_book** 🔗

Enable to allow access to contacts in the user's address book, if it's enabled you should also provide usage message in the privacy/address_book_usage_description option. See com.apple.security.personal-information.addressbook [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_personal-information_addressbook].

---

bool **codesign/entitlements/allow_dyld_environment_variables** 🔗

Allows app to use dynamic linker environment variables to inject code. If you are using add-ons with dynamic or self-modifying native code, enable them according to the add-on documentation. See com.apple.security.cs.allow-dyld-environment-variables [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_cs_allow-dyld-environment-variables].

---

bool **codesign/entitlements/allow_jit_code_execution** 🔗

Allows creating writable and executable memory for JIT code. If you are using add-ons with dynamic or self-modifying native code, enable them according to the add-on documentation. See com.apple.security.cs.allow-jit [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_cs_allow-jit].

---

bool **codesign/entitlements/allow_unsigned_executable_memory**

Allows creating writable and executable memory without JIT restrictions. If you are using add-ons with dynamic or self-modifying native code, enable them according to the add-on documentation. See com.apple.security.cs.allow-unsigned-executable-memory [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_cs_allow-unsigned-executable-memory].

---

bool **codesign/entitlements/app_sandbox/device_bluetooth**

Enable to allow app to interact with Bluetooth devices. This entitlement is required to use wireless controllers. See com.apple.security.device.bluetooth [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_device_bluetooth].

---

bool **codesign/entitlements/app_sandbox/device_usb**

Enable to allow app to interact with USB devices. This entitlement is required to use wired controllers. See com.apple.security.device.usb [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_device_usb].

---

bool **codesign/entitlements/app_sandbox/enabled**

Enables App Sandbox. The App Sandbox restricts access to user data, networking, and devices. Sandboxed apps can't access most of the file system, can't use custom file dialogs and execute binaries outside the .app bundle. See App Sandbox [https://developer.apple.com/documentation/security/app_sandbox].

Note: To distribute an app through the App Store, you must enable the App Sandbox.

---

int **codesign/entitlements/app_sandbox/files_downloads**

Allows read or write access to the user's "Downloads" folder. See com.apple.security.files.downloads.read-write [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_files_downloads_read-write].

---

int **codesign/entitlements/app_sandbox/files_movies** 🔗

Allows read or write access to the user's "Movies" folder. See com.apple.security.files.movies.read-write [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_assets_movies_read-write].

---

int **codesign/entitlements/app_sandbox/files_music** 🔗

Allows read or write access to the user's "Music" folder. See com.apple.security.files.music.read-write [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_assets_music_read-write].

---

int **codesign/entitlements/app_sandbox/files_pictures** 🔗

Allows read or write access to the user's "Pictures" folder. See com.apple.security.files.pictures.read-write [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_assets_pictures_read-write].

---

int **codesign/entitlements/app_sandbox/files_user_selected** 🔗

Allows read or write access to the locations the user has selected using a native file dialog. See com.apple.security.files.user-selected.read-write [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_files_user-selected_read-write].

Array **codesign/entitlements/app_sandbox/helper_executables** 🔗

List of helper executables to embedded to the app bundle. Sandboxed app are limited to execute only these executable. See Embedding a command-line tool in a sandboxed app [https://developer.apple.com/documentation/xcode/embedding-a-helper-tool-in-a-sandboxed-app].

---

bool **codesign/entitlements/app_sandbox/network_client** 🔗

Enable to allow app to establish outgoing network connections. See com.apple.security.network.client [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_network_client].

---

bool **codesign/entitlements/app_sandbox/network_server** 🔗

Enable to allow app to listen for incoming network connections. See com.apple.security.network.server [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_network_server].

bool **codesign/entitlements/apple_events** 🔗

Enable to allow app to send Apple events to other apps. See com.apple.security.automation.apple-events [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_automation_apple-events].

bool **codesign/entitlements/audio_input** 🔗

Enable if you need to use the microphone or other audio input sources, if it's enabled you should also provide usage message in the privacy/microphone_usage_description option. See com.apple.security.device.audio-input [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_device_audio-input].

bool **codesign/entitlements/calendars** 🔗

Enable to allow access to the user's calendar, if it's enabled you should also provide usage message in the privacy/calendar_usage_description option. See com.apple.security.personal-information.calendars [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_personal-information_calendars].

bool **codesign/entitlements/camera** 🔗

Enable if you need to use the camera, if it's enabled you should also provide usage message in the privacy/camera_usage_description option. See com.apple.security.device.camera [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_device_camera].

String **codesign/entitlements/custom_file** 🔗

Custom entitlements `.plist` file, if specified the rest of entitlements in the export config are ignored.

---

bool **codesign/entitlements/debugging** 🔗

You can temporarily enable this entitlement to use native debugger (GDB, LLDB) with the exported app. This entitlement should be disabled for production export. See Embedding a command-line tool in a sandboxed app [https://developer.apple.com/documentation/xcode/embedding-a-helper-tool-in-a-sandboxed-app].

---

bool **codesign/entitlements/disable_library_validation** 🔗

Allows app to load arbitrary libraries and frameworks (not signed with the same Team ID as the main executable or by Apple). Enable it if you are using GDExtension add-ons or ad-hoc signing, or want to support user-provided external add-ons. See com.apple.security.cs.disable-library-validation [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_cs_disable-library-validation].

---

bool **codesign/entitlements/location** 🔗

Enable if you need to use location information from Location Services, if it's enabled you should also provide usage message in the privacy/location_usage_description option. See com.apple.security.personal-information.location [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_personal-information_location].

---

bool **codesign/entitlements/photos_library** 🔗

Enable to allow access to the user's Photos library, if it's enabled you should also provide usage message in the privacy/photos_library_usage_description option. See com.apple.security.personal-information.photos-library [https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_personal-information_photos-library].

String **codesign/identity** 🔗

The "Full Name", "Common Name" or SHA-1 hash of the signing identity used to sign `.app` bundle.

String **codesign/installer_identity** 🔗

The "Full Name", "Common Name" or SHA-1 hash of the signing identity used to sign `.pkg` installer package for App Store distribution, use `3rd Party Mac Developer`

`Installer: Name.` identity.

String **codesign/provisioning_profile** 🔗

Provisioning profile file downloaded from Apple developer account dashboard. See Edit, download, or delete provisioning profiles [https://developer.apple.com/help/account/manage-profiles/edit-download-or-delete-profiles].

Can be overridden with the environment variable `GODOT_MACOS_CODESIGN_PROVISIONING_PROFILE`.

String **custom_template/debug** 🔗

Path to the custom export template. If left empty, default template is used.

String **custom_template/release** 🔗

Path to the custom export template. If left empty, default template is used.

---

int **debug/export_console_wrapper** 🔗

If enabled, a wrapper that can be used to run the application with console output is created alongside the exported application.

---

bool **display/high_res** 🔗

If `true`, the application is rendered at native display resolution, otherwise it is always rendered at loDPI resolution and upscaled by OS when required.

---

int **export/distribution_type** 🔗

Application distribution target.

String **notarization/api_key** 🔗

Apple App Store Connect API issuer key file.

Can be overridden with the environment variable `GODOT_MACOS_NOTARIZATION_API_KEY`.

String **notarization/api_key_id** 🔗

Apple App Store Connect API issuer key ID.

Can be overridden with the environment variable `GODOT_MACOS_NOTARIZATION_API_KEY_ID`.

String **notarization/api_uuid** 🔗

Apple App Store Connect API issuer UUID.

Can be overridden with the environment variable `GODOT_MACOS_NOTARIZATION_API_UUID`.

String **notarization/apple_id_name** 🔗

Apple ID account name (email address).

Can be overridden with the environment variable `GODOT_MACOS_NOTARIZATION_APPLE_ID_NAME`.

String **notarization/apple_id_password** 🔗

Apple ID app-specific password.

Can be overridden with the environment variable `GODOT_MACOS_NOTARIZATION_APPLE_ID_PASSWORD`.

---

int **notarization/notarization** 🔗

Tool to use for notarization.

String **privacy/address_book_usage_description** 🔗

A message displayed when requesting access to the user's contacts (in English).

Dictionary **privacy/address_book_usage_description_localized** 🔗

A message displayed when requesting access to the user's contacts (localized).

String **privacy/calendar_usage_description** 🔗

A message displayed when requesting access to the user's calendar data (in English).

Dictionary **privacy/calendar_usage_description_localized** 🔗

A message displayed when requesting access to the user's calendar data (localized).

String **privacy/camera_usage_description** 🔗

A message displayed when requesting access to the device's camera (in English).

Dictionary **privacy/camera_usage_description_localized** 🔗

A message displayed when requesting access to the device's camera (localized).

bool **privacy/collected_data/advertising_data/collected** 🔗

Indicates whether your app collects advertising data.

int **privacy/collected_data/advertising_data/collection_purposes** 🔗

The reasons your app collects advertising data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/advertising_data/linked_to_user** 🔗

Indicates whether your app links advertising data to the user's identity.

bool **privacy/collected_data/advertising_data/used_for_tracking** 🔗

Indicates whether your app uses advertising data for tracking.

bool **privacy/collected_data/audio_data/collected** 🔗

Indicates whether your app collects audio data.

int **privacy/collected_data/audio_data/collection_purposes** 🔗

The reasons your app collects audio data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/audio_data/linked_to_user** 🔗

Indicates whether your app links audio data to the user's identity.

bool **privacy/collected_data/audio_data/used_for_tracking** 🔗

Indicates whether your app uses audio data for tracking.

bool **privacy/collected_data/browsing_history/collected** 🔗

Indicates whether your app collects browsing history.

int **privacy/collected_data/browsing_history/collection_purposes** 🔗

The reasons your app collects browsing history. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/browsing_history/linked_to_user** 🔗

Indicates whether your app links browsing history to the user's identity.

bool **privacy/collected_data/browsing_history/used_for_tracking** 🔗

Indicates whether your app uses browsing history for tracking.

bool **privacy/collected_data/coarse_location/collected** 🔗

Indicates whether your app collects coarse location data.

int **privacy/collected_data/coarse_location/collection_purposes** 🔗

The reasons your app collects coarse location data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/coarse_location/linked_to_user**

Indicates whether your app links coarse location data to the user's identity.

bool **privacy/collected_data/coarse_location/used_for_tracking**

Indicates whether your app uses coarse location data for tracking.

bool **privacy/collected_data/contacts/collected**

Indicates whether your app collects contacts.

int **privacy/collected_data/contacts/collection_purposes**

The reasons your app collects contacts. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/contacts/linked_to_user**

Indicates whether your app links contacts to the user's identity.

bool **privacy/collected_data/contacts/used_for_tracking**

Indicates whether your app uses contacts for tracking.

bool **privacy/collected_data/crash_data/collected**

Indicates whether your app collects crash data.

int **privacy/collected_data/crash_data/collection_purposes**

The reasons your app collects crash data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/crash_data/linked_to_user**

Indicates whether your app links crash data to the user's identity.

bool **privacy/collected_data/crash_data/used_for_tracking** 🔗

Indicates whether your app uses crash data for tracking.

bool **privacy/collected_data/credit_info/collected** 🔗

Indicates whether your app collects credit information.

int **privacy/collected_data/credit_info/collection_purposes** 🔗

The reasons your app collects credit information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/credit_info/linked_to_user** 🔗

Indicates whether your app links credit information to the user's identity.

bool **privacy/collected_data/credit_info/used_for_tracking** 🔗

Indicates whether your app uses credit information for tracking.

bool **privacy/collected_data/customer_support/collected** 🔗

Indicates whether your app collects customer support data.

int **privacy/collected_data/customer_support/collection_purposes** 🔗

The reasons your app collects customer support data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/customer_support/linked_to_user** 🔗

Indicates whether your app links customer support data to the user's identity.

bool **privacy/collected_data/customer_support/used_for_tracking** 🔗

Indicates whether your app uses customer support data for tracking.

bool **privacy/collected_data/device_id/collected**

Indicates whether your app collects device IDs.

int **privacy/collected_data/device_id/collection_purposes**

The reasons your app collects device IDs. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/device_id/linked_to_user**

Indicates whether your app links device IDs to the user's identity.

bool **privacy/collected_data/device_id/used_for_tracking**

Indicates whether your app uses device IDs for tracking.

bool **privacy/collected_data/email_address/collected**

Indicates whether your app collects email address.

int **privacy/collected_data/email_address/collection_purposes**

The reasons your app collects email address. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/email_address/linked_to_user**

Indicates whether your app links email address to the user's identity.

bool **privacy/collected_data/email_address/used_for_tracking**

Indicates whether your app uses email address for tracking.

bool **privacy/collected_data/emails_or_text_messages/collected** 🔗

Indicates whether your app collects emails or text messages.

---

int **privacy/collected_data/emails_or_text_messages/collection_purposes** 🔗

The reasons your app collects emails or text messages. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/emails_or_text_messages/linked_to_user** 🔗

Indicates whether your app links emails or text messages to the user's identity.

---

bool **privacy/collected_data/emails_or_text_messages/used_for_tracking** 🔗

Indicates whether your app uses emails or text messages for tracking.

---

bool **privacy/collected_data/environment_scanning/collected** 🔗

Indicates whether your app collects environment scanning data.

---

int **privacy/collected_data/environment_scanning/collection_purposes** 🔗

The reasons your app collects environment scanning data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/environment_scanning/linked_to_user** 🔗

Indicates whether your app links environment scanning data to the user's identity.

---

bool **privacy/collected_data/environment_scanning/used_for_tracking** 🔗

Indicates whether your app uses environment scanning data for tracking.

---

bool **privacy/collected_data/fitness/collected** 🔗

Indicates whether your app collects fitness and exercise data.

int **privacy/collected_data/fitness/collection_purposes**

The reasons your app collects fitness and exercise data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/fitness/linked_to_user**

Indicates whether your app links fitness and exercise data to the user's identity.

bool **privacy/collected_data/fitness/used_for_tracking**

Indicates whether your app uses fitness and exercise data for tracking.

bool **privacy/collected_data/gameplay_content/collected**

Indicates whether your app collects gameplay content.

int **privacy/collected_data/gameplay_content/collection_purposes**

The reasons your app collects gameplay content. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/gameplay_content/linked_to_user**

Indicates whether your app links gameplay content to the user's identity.

bool **privacy/collected_data/gameplay_content/used_for_tracking**

Indicates whether your app uses gameplay content for tracking.

bool **privacy/collected_data/hands/collected**

Indicates whether your app collects user's hand structure and hand movements.

int **privacy/collected_data/hands/collection_purposes**

The reasons your app collects user's hand structure and hand movements. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/hands/linked_to_user**

Indicates whether your app links user's hand structure and hand movements to the user's identity.

bool **privacy/collected_data/hands/used_for_tracking**

Indicates whether your app uses user's hand structure and hand movements for tracking.

bool **privacy/collected_data/head/collected**

Indicates whether your app collects user's head movement.

int **privacy/collected_data/head/collection_purposes**

The reasons your app collects user's head movement. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/head/linked_to_user**

Indicates whether your app links user's head movement to the user's identity.

bool **privacy/collected_data/head/used_for_tracking**

Indicates whether your app uses user's head movement for tracking.

bool **privacy/collected_data/health/collected**

Indicates whether your app collects health and medical data.

int **privacy/collected_data/health/collection_purposes** 🔗

The reasons your app collects health and medical data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/health/linked_to_user** 🔗

Indicates whether your app links health and medical data to the user's identity.

---

bool **privacy/collected_data/health/used_for_tracking** 🔗

Indicates whether your app uses health and medical data for tracking.

---

bool **privacy/collected_data/name/collected** 🔗

Indicates whether your app collects user's name.

---

int **privacy/collected_data/name/collection_purposes** 🔗

The reasons your app collects user's name. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/name/linked_to_user** 🔗

Indicates whether your app links user's name to the user's identity.

---

bool **privacy/collected_data/name/used_for_tracking** 🔗

Indicates whether your app uses user's name for tracking.

---

bool **privacy/collected_data/other_contact_info/collected** 🔗

Indicates whether your app collects any other contact information.

---

int **privacy/collected_data/other_contact_info/collection_purposes** 🔗

The reasons your app collects any other contact information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/other_contact_info/linked_to_user**

Indicates whether your app links any other contact information to the user's identity.

bool **privacy/collected_data/other_contact_info/used_for_tracking**

Indicates whether your app uses any other contact information for tracking.

bool **privacy/collected_data/other_data_types/collected**

Indicates whether your app collects any other data.

int **privacy/collected_data/other_data_types/collection_purposes**

The reasons your app collects any other data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/other_data_types/linked_to_user**

Indicates whether your app links any other data to the user's identity.

bool **privacy/collected_data/other_data_types/used_for_tracking**

Indicates whether your app uses any other data for tracking.

bool **privacy/collected_data/other_diagnostic_data/collected**

Indicates whether your app collects any other diagnostic data.

int **privacy/collected_data/other_diagnostic_data/collection_purposes**

The reasons your app collects any other diagnostic data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/other_diagnostic_data/linked_to_user**

Indicates whether your app links any other diagnostic data to the user's identity.

---

bool **privacy/collected_data/other_diagnostic_data/used_for_tracking**

Indicates whether your app uses any other diagnostic data for tracking.

---

bool **privacy/collected_data/other_financial_info/collected**

Indicates whether your app collects any other financial information.

---

int **privacy/collected_data/other_financial_info/collection_purposes**

The reasons your app collects any other financial information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/other_financial_info/linked_to_user**

Indicates whether your app links any other financial information to the user's identity.

---

bool **privacy/collected_data/other_financial_info/used_for_tracking**

Indicates whether your app uses any other financial information for tracking.

---

bool **privacy/collected_data/other_usage_data/collected**

Indicates whether your app collects any other usage data.

---

int **privacy/collected_data/other_usage_data/collection_purposes**

The reasons your app collects any other usage data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/other_usage_data/linked_to_user**

Indicates whether your app links any other usage data to the user's identity.

bool **privacy/collected_data/other_usage_data/used_for_tracking**

Indicates whether your app uses any other usage data for tracking.

bool **privacy/collected_data/other_user_content/collected**

Indicates whether your app collects any other user generated content.

int **privacy/collected_data/other_user_content/collection_purposes**

The reasons your app collects any other user generated content. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/other_user_content/linked_to_user**

Indicates whether your app links any other user generated content to the user's identity.

bool **privacy/collected_data/other_user_content/used_for_tracking**

Indicates whether your app uses any other user generated content for tracking.

bool **privacy/collected_data/payment_info/collected**

Indicates whether your app collects payment information.

int **privacy/collected_data/payment_info/collection_purposes**

The reasons your app collects payment information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/payment_info/linked_to_user**

Indicates whether your app links payment information to the user's identity.

---

bool **privacy/collected_data/payment_info/used_for_tracking**

Indicates whether your app uses payment information for tracking.

---

bool **privacy/collected_data/performance_data/collected**

Indicates whether your app collects performance data.

---

int **privacy/collected_data/performance_data/collection_purposes**

The reasons your app collects performance data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/performance_data/linked_to_user**

Indicates whether your app links performance data to the user's identity.

---

bool **privacy/collected_data/performance_data/used_for_tracking**

Indicates whether your app uses performance data for tracking.

---

bool **privacy/collected_data/phone_number/collected**

Indicates whether your app collects phone number.

---

int **privacy/collected_data/phone_number/collection_purposes**

The reasons your app collects phone number. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/phone_number/linked_to_user**

Indicates whether your app links phone number to the user's identity.

---

bool **privacy/collected_data/phone_number/used_for_tracking** 🔗

Indicates whether your app uses phone number for tracking.

---

bool **privacy/collected_data/photos_or_videos/collected** 🔗

Indicates whether your app collects photos or videos.

---

int **privacy/collected_data/photos_or_videos/collection_purposes** 🔗

The reasons your app collects photos or videos. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/photos_or_videos/linked_to_user** 🔗

Indicates whether your app links photos or videos to the user's identity.

---

bool **privacy/collected_data/photos_or_videos/used_for_tracking** 🔗

Indicates whether your app uses photos or videos for tracking.

---

bool **privacy/collected_data/physical_address/collected** 🔗

Indicates whether your app collects physical address.

---

int **privacy/collected_data/physical_address/collection_purposes** 🔗

The reasons your app collects physical address. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

---

bool **privacy/collected_data/physical_address/linked_to_user** 🔗

Indicates whether your app links physical address to the user's identity.

---

bool **privacy/collected_data/physical_address/used_for_tracking**

Indicates whether your app uses physical address for tracking.

bool **privacy/collected_data/precise_location/collected**

Indicates whether your app collects precise location data.

int **privacy/collected_data/precise_location/collection_purposes**

The reasons your app collects precise location data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/precise_location/linked_to_user**

Indicates whether your app links precise location data to the user's identity.

bool **privacy/collected_data/precise_location/used_for_tracking**

Indicates whether your app uses precise location data for tracking.

bool **privacy/collected_data/product_interaction/collected**

Indicates whether your app collects product interaction data.

int **privacy/collected_data/product_interaction/collection_purposes**

The reasons your app collects product interaction data. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/product_interaction/linked_to_user**

Indicates whether your app links product interaction data to the user's identity.

bool **privacy/collected_data/product_interaction/used_for_tracking**

Indicates whether your app uses product interaction data for tracking.

bool **privacy/collected_data/purchase_history/collected** 🔗

Indicates whether your app collects purchase history.

int **privacy/collected_data/purchase_history/collection_purposes** 🔗

The reasons your app collects purchase history. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/purchase_history/linked_to_user** 🔗

Indicates whether your app links purchase history to the user's identity.

bool **privacy/collected_data/purchase_history/used_for_tracking** 🔗

Indicates whether your app uses purchase history for tracking.

bool **privacy/collected_data/search_hhistory/collected** 🔗

Indicates whether your app collects search history.

int **privacy/collected_data/search_hhistory/collection_purposes** 🔗

The reasons your app collects search history. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/search_hhistory/linked_to_user** 🔗

Indicates whether your app links search history to the user's identity.

bool **privacy/collected_data/search_hhistory/used_for_tracking** 🔗

Indicates whether your app uses search history for tracking.

bool **privacy/collected_data/sensitive_info/collected**

Indicates whether your app collects sensitive user information.

int **privacy/collected_data/sensitive_info/collection_purposes**

The reasons your app collects sensitive user information. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/sensitive_info/linked_to_user**

Indicates whether your app links sensitive user information to the user's identity.

bool **privacy/collected_data/sensitive_info/used_for_tracking**

Indicates whether your app uses sensitive user information for tracking.

bool **privacy/collected_data/user_id/collected**

Indicates whether your app collects user IDs.

int **privacy/collected_data/user_id/collection_purposes**

The reasons your app collects user IDs. See Describing data use in privacy manifests [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_data_use_in_privacy_manifests].

bool **privacy/collected_data/user_id/linked_to_user**

Indicates whether your app links user IDs to the user's identity.

bool **privacy/collected_data/user_id/used_for_tracking**

Indicates whether your app uses user IDs for tracking.

String **privacy/desktop_folder_usage_description**

A message displayed when requesting access to the user's "Desktop" folder (in English).

Dictionary **privacy/desktop_folder_usage_description_localized**

A message displayed when requesting access to the user's "Desktop" folder (localized).

String **privacy/documents_folder_usage_description**

A message displayed when requesting access to the user's "Documents" folder (in English).

Dictionary **privacy/documents_folder_usage_description_localized**

A message displayed when requesting access to the user's "Documents" folder (localized).

String **privacy/downloads_folder_usage_description**

A message displayed when requesting access to the user's "Downloads" folder (in English).

Dictionary **privacy/downloads_folder_usage_description_localized**

A message displayed when requesting access to the user's "Downloads" folder (localized).

String **privacy/location_usage_description**

A message displayed when requesting access to the user's location information (in English).

Dictionary **privacy/location_usage_description_localized**

A message displayed when requesting access to the user's location information (localized).

String **privacy/microphone_usage_description**

A message displayed when requesting access to the device's microphone (in English).

Dictionary **privacy/microphone_usage_description_localized**

A message displayed when requesting access to the device's microphone (localized).

String **privacy/network_volumes_usage_description**

A message displayed when requesting access to the user's network drives (in English).

Dictionary **privacy/network_volumes_usage_description_localized**

A message displayed when requesting access to the user's network drives (localized).

String **privacy/photos_library_usage_description**

A message displayed when requesting access to the user's photo library (in English).

Dictionary **privacy/photos_library_usage_description_localized**

A message displayed when requesting access to the user's photo library (localized).

String **privacy/removable_volumes_usage_description**

A message displayed when requesting access to the user's removable drives (in English).

Dictionary **privacy/removable_volumes_usage_description_localized**

A message displayed when requesting access to the user's removable drives (localized).

PackedStringArray **privacy/tracking_domains**

The list of internet domains your app connects to that engage in tracking. See Privacy manifest files [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files].

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedStringArray for more details.

bool **privacy/tracking_enabled**

Indicates whether your app uses data for tracking. See Privacy manifest files [https://developer.apple.com/documentation/bundleresources/privacy_manifest_files].

String **ssh_remote_deploy/cleanup_script**

Script code to execute on the remote host when app is finished.

The following variables can be used in the script:

- `{temp_dir}` - Path of temporary folder on the remote, used to upload app and scripts to.
- `{archive_name}` - Name of the ZIP containing uploaded application.
- `{exe_name}` - Name of application executable.
- `{cmd_args}` - Array of the command line argument for the application.

---

bool **ssh_remote_deploy/enabled**

Enables remote deploy using SSH/SCP.

String **ssh_remote_deploy/extra_args_scp**

Array of the additional command line arguments passed to the SCP.

String **ssh_remote_deploy/extra_args_ssh**

Array of the additional command line arguments passed to the SSH.

String **ssh_remote_deploy/host**

Remote host SSH user name and address, in `user@address` format.

String **ssh_remote_deploy/port**

Remote host SSH port number.

String **ssh_remote_deploy/run_script**

Script code to execute on the remote host when running the app.

The following variables can be used in the script:

- `{temp_dir}` - Path of temporary folder on the remote, used to upload app and scripts to.
- `{archive_name}` - Name of the ZIP containing uploaded application.

- `{exe_name}` - Name of application executable.
- `{cmd_args}` - Array of the command line argument for the application.

String **xcode/platform_build**

macOS build number used to build application executable.

String **xcode/sdk_build**

macOS SDK build number used to build application executable.

String **xcode/sdk_name**

macOS SDK name used to build application executable.

String **xcode/sdk_version**

macOS SDK version used to build application executable in the `major.minor` format.

String **xcode/xcode_build**

Xcode build number used to build application executable.

String **xcode/xcode_version**

Xcode version used to build application executable.

EditorExportPlatformPC

Inherits: EditorExportPlatform **<** RefCounted **<** Object

Inherited By: EditorExportPlatformLinuxBSD, EditorExportPlatformWindows

Base class for the desktop platform exporter (Windows and Linux/BSD).

Description

The base class for the desktop platform exporters. These include Windows and Linux/BSD, but not macOS. See the classes inheriting this one for more details.

Tutorials

- Exporting for Windows
- Exporting for Linux

EditorExportPlatformWeb

Inherits: EditorExportPlatform **<** RefCounted **<** Object

Exporter for the Web.

Description

The Web exporter customizes how a web build is handled. In the editor's "Export" window, it is created when adding a new "Web" preset.

Note: Godot on Web is rendered inside a `<canvas>` tag. Normally, the canvas cannot be positioned or resized manually, but otherwise acts as the main Window of the application.

Tutorials

- Exporting for the Web
- Web documentation index

Properties

String	custom_template/debug
String	custom_template/release
int	html/canvas_resize_policy

String	html/custom_html_shell
bool	html/experimental_virtual_keyboard
bool	html/export_icon
bool	html/focus_canvas_on_start
String	html/head_include
Color	progressive_web_app/background_color
int	progressive_web_app/display
bool	progressive_web_app/enabled
bool	progressive_web_app/ensure_cross_origin_isolation_headers
String	progressive_web_app/icon_144x144
String	progressive_web_app/icon_180x180
String	progressive_web_app/icon_512x512
String	progressive_web_app/offline_page

int	progressive_web_app/orientation
bool	variant/extensions_support
bool	variant/thread_support
bool	vram_texture_compression/for_desktop
bool	vram_texture_compression/for_mobile

Property Descriptions

String **custom_template/debug** 🔗

File path to the custom export template used for debug builds. If left empty, the default template is used.

String **custom_template/release** 🔗

File path to the custom export template used for release builds. If left empty, the default template is used.

int **html/canvas_resize_policy** 🔗

Determines how the canvas should be resized by Godot.

- **None:** The canvas is not automatically resized.

- **Project:** The size of the canvas is dependent on the ProjectSettings.
- **Adaptive:** The canvas is automatically resized to fit as much of the web page as possible.

String **html/custom_html_shell**

The custom HTML page that wraps the exported web build. If left empty, the default HTML shell is used.

For more information, see the Customizing HTML5 Shell tutorial.

---

bool **html/experimental_virtual_keyboard**

Experimental: This property may be changed or removed in future versions.

If `true`, embeds support for a virtual keyboard into the web page, which is shown when necessary on touchscreen devices.

---

bool **html/export_icon**

If `true`, the project icon will be used as the favicon for this application's web page.

---

bool **html/focus_canvas_on_start**

If `true`, the canvas will be focused as soon as the application is loaded, if the browser window is already in focus.

String **html/head_include**

Additional HTML tags to include inside the `<head>`, such as `<meta>` tags.

Note: You do not need to add a `<title>` tag, as it is automatically included based on the project's name.

Color **progressive_web_app/background_color** 🔗

The background color used behind the web application.

---

int **progressive_web_app/display** 🔗

The display mode [https://developer.mozilla.org/en-US/docs/Web/Manifest/display/] to use for this progressive web application. Different browsers and platforms may not behave the same.

- **Fullscreen:** Displays the app in fullscreen and hides all of the browser's UI elements.
- **Standalone:** Displays the app in a separate window and hides all of the browser's UI elements.
- **Minimal UI:** Displays the app in a separate window and only shows the browser's UI elements for navigation.
- **Browser:** Displays the app as a normal web page.

---

bool **progressive_web_app/enabled** 🔗

If `true`, turns this web build into a progressive web application [https://en.wikipedia.org/wiki/Progressive_web_app] (PWA).

---

bool **progressive_web_app/ensure_cross_origin_isolation_heade**

rs 🔗

When enabled, the progressive web app will make sure that each request has cross-origin isolation headers (COEP/COOP).

This can simplify the setup to serve the exported game.

String **progressive_web_app/icon_144x144** 🔗

File path to the smallest icon for this web application. If not defined, defaults to the project icon.

Note: If the icon is not 144x144, it will be automatically resized for the final build.

String **progressive_web_app/icon_180x180** 🔗

File path to the small icon for this web application. If not defined, defaults to the project icon.

Note: If the icon is not 180x180, it will be automatically resized for the final build.

String **progressive_web_app/icon_512x512** 🔗

File path to the smallest icon for this web application. If not defined, defaults to the project icon.

Note: If the icon is not 512x512, it will be automatically resized for the final build.

String **progressive_web_app/offline_page** 🔗

The page to display, should the server hosting the page not be available. This page is saved in the client's machine.

---

int **progressive_web_app/orientation** 🔗

The orientation to use when the web application is run through a mobile device.

- **Any:** No orientation is forced.
- **Landscape:** Forces a horizontal layout (wider than it is taller).
- **Portrait:** Forces a vertical layout (taller than it is wider).

---

bool **variant/extensions_support** 🔗

If `true` enables GDExtension support for this web build.

---

bool **variant/thread_support** 🔗

If `true`, the exported game will support threads. It requires a "cross-origin isolated" website [https://web.dev/articles/coop-coep], which may be difficult to set up and is limited for security reasons (such as not being able to communicate with third-party websites).

If `false`, the exported game will not support threads. As a result, it is more prone to performance and audio issues, but will only require to be run on an HTTPS website.

---

bool **vram_texture_compression/for_desktop** 🔗

If `true`, allows textures to be optimized for desktop through the S3TC algorithm.

---

bool **vram_texture_compression/for_mobile** 🔗

If `true` allows textures to be optimized for mobile through the ETC2 algorithm.

EditorExportPlatformWindows

Inherits: EditorExportPlatformPC **<** EditorExportPlatform **<** RefCounted **<** Object

Exporter for Windows.

Description

The Windows exporter customizes how a Windows build is handled. In the editor's "Export" window, it is created when adding a new "Windows" preset.

Tutorials

- Exporting for Windows

Properties

String	application/company_name
String	application/console_wrapper_icon
String	application/copyright

bool	application/d3d12_agility_sdk_multiarch
int	application/export_angle
int	application/export_d3d12
String	application/file_description
String	application/file_version
String	application/icon
int	application/icon_interpolation
bool	application/modify_resources
String	application/product_name
String	application/product_version
String	application/trademarks
String	binary_format/architecture
bool	binary_format/embed_pck

PackedStringArray	codesign/custom_options
String	codesign/description
int	codesign/digest_algorithm
bool	codesign/enable
String	codesign/identity
int	codesign/identity_type
String	codesign/password
bool	codesign/timestamp
String	codesign/timestamp_server_url
String	custom_template/debug
String	custom_template/release
int	debug/export_console_wrapper
String	ssh_remote_deploy/cleanup_script

bool	ssh_remote_deploy/enabled
String	ssh_remote_deploy/extra_args_scp
String	ssh_remote_deploy/extra_args_ssh
String	ssh_remote_deploy/host
String	ssh_remote_deploy/port
String	ssh_remote_deploy/run_script
bool	texture_format/etc2_astc
bool	texture_format/s3tc_bptc

Property Descriptions

String **application/company_name** 🔗

Company that produced the application. Required. See StringFileInfo [https://learn.microsoft.com/en-us/windows/win32/menurc/stringfileinfo-block].

String **application/console_wrapper_icon** 🔗

Console wrapper icon file. If left empty, it will fallback to application/icon, then to ProjectSettings.application/config/windows_native_icon, and lastly, ProjectSettings.application/config/icon.

String **application/copyright**

Copyright notice for the bundle visible to the user. Optional. See StringFileInfo [https://learn.microsoft.com/en-us/windows/win32/menurc/stringfileinfo-block].

---

bool **application/d3d12_agility_sdk_multiarch**

If `true`, and application/export_d3d12 is set, the Agility SDK DLLs will be stored in arch-specific subdirectories.

---

int **application/export_angle**

If set to `1`, ANGLE libraries are exported with the exported application. If set to `0`, ANGLE libraries are exported only if ProjectSettings.rendering/gl_compatibility/driver is set to `"opengl3_angle"`.

---

int **application/export_d3d12**

If set to `1`, the Direct3D 12 runtime libraries (Agility SDK, PIX) are exported with the exported application. If set to `0`, Direct3D 12 libraries are exported only if ProjectSettings.rendering/rendering_device/driver is set to `"d3d12"`.

String **application/file_description**

File description to be presented to users. Required. See StringFileInfo [https://learn.microsoft.com/en-us/windows/win32/menurc/stringfileinfo-block].

String **application/file_version** 🔗

Version number of the file. Falls back to ProjectSettings.application/config/version if left empty. See StringFileInfo [https://learn.microsoft.com/en-us/windows/win32/menurc/stringfileinfo-block].

String **application/icon** 🔗

Application icon file. If left empty, it will fallback to ProjectSettings.application/config/windows_native_icon, and then to ProjectSettings.application/config/icon.

---

int **application/icon_interpolation** 🔗

Interpolation method used to resize application icon.

---

bool **application/modify_resources** 🔗

If enabled, icon and metadata of the exported executable is set according to the other `application/*` values.

String **application/product_name** 🔗

Name of the application. Required. See StringFileInfo [https://learn.microsoft.com/en-us/windows/win32/menurc/stringfileinfo-block].

String **application/product_version** 🔗

Application version visible to the user. Falls back to ProjectSettings.application/config/version if left empty. See StringFileInfo [https://learn.microsoft.com/en-us/windows/win32/menurc/stringfileinfo-block].

String **application/trademarks** 🔗

Trademarks and registered trademarks that apply to the file. Optional. See StringFileInfo [https://learn.microsoft.com/en-us/windows/win32/menurc/stringfileinfo-block].

String **binary_format/architecture** 🔗

Application executable architecture.

Supported architectures: `x86_32`, `x86_64`, and `arm64`.

Official export templates include `x86_32` and `x86_64` binaries only.

bool **binary_format/embed_pck** 🔗

If `true`, project resources are embedded into the executable.

PackedStringArray **codesign/custom_options** 🔗

Array of the additional command line arguments passed to the code signing tool. See Sign Tool [https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe].

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedStringArray for more details.

String **codesign/description** 🔗

Description of the signed content. See Sign Tool [https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe].

---

int **codesign/digest_algorithm** 🔗

Digest algorithm to use for creating signature. See Sign Tool [https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe].

---

bool **codesign/enable** 🔗

If `true`, executable signing is enabled.

String **codesign/identity** 🔗

PKCS #12 certificate file used to sign executable or certificate SHA-1 hash (if codesign/identity_type is set to "Use certificate store"). See Sign Tool [https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe].

Can be overridden with the environment variable `GODOT_WINDOWS_CODESIGN_IDENTITY`.

---

int **codesign/identity_type** 🔗

Type of identity to use. See Sign Tool [https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe].

Can be overridden with the environment variable `GODOT_WINDOWS_CODESIGN_IDENTITY_TYPE`.

String **codesign/password** 🔗

Password for the certificate file used to sign executable. See Sign Tool [https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe].

Can be overridden with the environment variable `GODOT_WINDOWS_CODESIGN_PASSWORD`.

---

bool **codesign/timestamp** 🔗

If `true`, time-stamp is added to the signature. See Sign Tool [https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe].

String **codesign/timestamp_server_url** 🔗

URL of the time stamp server. If left empty, the default server is used. See Sign Tool [https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe].

String **custom_template/debug** 🔗

Path to the custom export template. If left empty, default template is used.

String **custom_template/release** 🔗

Path to the custom export template. If left empty, default template is used.

---

int **debug/export_console_wrapper**

If `true`, a console wrapper executable is exported alongside the main executable, which allows running the project with enabled console output.

String **ssh_remote_deploy/cleanup_script**

Script code to execute on the remote host when app is finished.

The following variables can be used in the script:

- `{temp_dir}` - Path of temporary folder on the remote, used to upload app and scripts to.
- `{archive_name}` - Name of the ZIP containing uploaded application.
- `{exe_name}` - Name of application executable.
- `{cmd_args}` - Array of the command line argument for the application.

---

bool **ssh_remote_deploy/enabled**

Enables remote deploy using SSH/SCP.

String **ssh_remote_deploy/extra_args_scp**

Array of the additional command line arguments passed to the SCP.

String **ssh_remote_deploy/extra_args_ssh**

Array of the additional command line arguments passed to the SSH.

String **ssh_remote_deploy/host**

Remote host SSH user name and address, in `user@address` format.

String **ssh_remote_deploy/port**

Remote host SSH port number.

String **ssh_remote_deploy/run_script**

Script code to execute on the remote host when running the app.

The following variables can be used in the script:

- `{temp_dir}` - Path of temporary folder on the remote, used to upload app and scripts to.
- `{archive_name}` - Name of the ZIP containing uploaded application.
- `{exe_name}` - Name of application executable.
- `{cmd_args}` - Array of the command line argument for the application.

---

bool **texture_format/etc2_astc**

If `true`, project textures are exported in the ETC2/ASTC format.

---

bool **texture_format/s3tc_bptc**

If `true`, project textures are exported in the S3TC/BPTC format.

EditorExportPlugin

Inherits: RefCounted **<** Object

A script that is executed when exporting the project.

Description

EditorExportPlugins are automatically invoked whenever the user exports the project. Their most common use is to determine what files are being included in the exported project. For each plugin, _export_begin is called at the beginning of the export process and then _export_file is called for each exported file.

To use **EditorExportPlugin**, register it using the EditorPlugin.add_export_plugin method first.

Tutorials

- Export Android plugins

Methods

bool	_begin_customize_resources(platform: EditorExportPlatform, features: PackedStringArray) virtual const
bool	_begin_customize_scenes(platform: EditorExportPlatform, features: PackedStringArray) virtual const
Resource	_customize_resource(resource: Resource, path: String) virtual
Node	_customize_scene(scene: Node, path: String) virtual
void	_end_customize_resources() virtual

void	_end_customize_scenes() virtual
void	_export_begin(features: PackedStringArray, is_debug: bool, path: String, flags: int) virtual
void	_export_end() virtual
void	_export_file(path: String, type: String, features: PackedStringArray) virtual
PackedStringArray	_get_android_dependencies(platform: EditorExportPlatform, debug: bool) virtual const
PackedStringArray	_get_android_dependencies_maven_repos(platform: EditorExportPlatform, debug: bool) virtual const
PackedStringArray	_get_android_libraries(platform: EditorExportPlatform, debug: bool) virtual const
String	_get_android_manifest_activity_element_contents(platform: EditorExportPlatform, debug: bool) virtual const
String	_get_android_manifest_application_element_contents(platform: EditorExportPlatform, debug: bool) virtual const
String	_get_android_manifest_element_contents(platform: EditorExportPlatform, debug: bool) virtual const
int	_get_customization_configuration_hash() virtual const
PackedStringArray	_get_export_features(platform: EditorExportPlatform, debug: bool) virtual const

String	_get_export_option_warning(platform: EditorExportPlatform, option: String) virtual const
Array[Dictionary]	_get_export_options(platform: EditorExportPlatform) virtual const
Dictionary	_get_export_options_overrides(platform: EditorExportPlatform) virtual const
String	_get_name() virtual const
bool	_should_update_export_options(platform: EditorExportPlatform) virtual const
bool	_supports_platform(platform: EditorExportPlatform) virtual const
void	add_file(path: String, file: PackedByteArray, remap: bool)
void	add_ios_bundle_file(path: String)
void	add_ios_cpp_code(code: String)
void	add_ios_embedded_framework(path: String)
void	add_ios_framework(path: String)
void	add_ios_linker_flags(flags: String)
void	add_ios_plist_content(plist_content: String)
void	add_ios_project_static_lib(path: String)

void	add_macos_plugin_file(path: String)
void	add_shared_object(path: String, tags: PackedStringArray, target: String)
Variant	get_option(name: StringName) const
void	skip()

Method Descriptions

bool **_begin_customize_resources**(platform: EditorExportPlatform, features: PackedStringArray) virtual const

Return `true` if this plugin will customize resources based on the platform and features used.

When enabled, _get_customization_configuration_hash and _customize_resource will be called and must be implemented.

---

bool **_begin_customize_scenes**(platform: EditorExportPlatform, features: PackedStringArray) virtual const

Return `true` if this plugin will customize scenes based on the platform and features used.

When enabled, _get_customization_configuration_hash and _customize_scene will be called and must be implemented.

Resource **_customize_resource**(resource: Resource, path: String) virtual

Customize a resource. If changes are made to it, return the same or a new resource. Otherwise, return `null`.

The *path* argument is only used when customizing an actual file, otherwise this means that this resource is part of another one and it will be empty.

Implementing this method is required if _begin_customize_resources returns `true`.

Node **_customize_scene**(scene: Node, path: String) virtual

Customize a scene. If changes are made to it, return the same or a new scene. Otherwise, return `null`. If a new scene is returned, it is up to you to dispose of the old one.

Implementing this method is required if _begin_customize_scenes returns `true`.

---

void **_end_customize_resources**() virtual

This is called when the customization process for resources ends.

---

void **_end_customize_scenes**() virtual

This is called when the customization process for scenes ends.

---

void **_export_begin**(features: PackedStringArray, is_debug: bool, path: String, flags: int) virtual

Virtual method to be overridden by the user. It is called when the export starts and provides all information about the export. `features` is the list of features for the export, `is_debug` is `true` for debug builds, `path` is the target path for the exported project. `flags` is only used when running a runnable profile, e.g. when using native run on Android.

---

void **_export_end**() virtual

Virtual method to be overridden by the user. Called when the export is finished.

---

void **_export_file**(path: String, type: String, features: PackedStringArray) virtual

Virtual method to be overridden by the user. Called for each exported file before _customize_resource and _customize_scene. The arguments can be used to identify the file. `path` is the path of the file, `type` is the Resource represented by the file (e.g. PackedScene), and `features` is the list of features for the export.

Calling skip inside this callback will make the file not included in the export.

PackedStringArray **_get_android_dependencies**(platform: EditorExportPlatform, debug: bool) virtual const 🔗

Virtual method to be overridden by the user. This is called to retrieve the set of Android dependencies provided by this plugin. Each returned Android dependency should have the format of an Android remote binary dependency: `org.godot.example:my-plugin:0.0.0`

For more information see Android documentation on dependencies [https://developer.android.com/build/dependencies?agpversion=4.1#dependency-types].

Note: Only supported on Android and requires EditorExportPlatformAndroid.gradle_build/use_gradle_build to be enabled.

PackedStringArray **_get_android_dependencies_maven_repos**(platform: EditorExportPlatform, debug: bool) virtual const 🔗

Virtual method to be overridden by the user. This is called to retrieve the URLs of Maven repositories for the set of Android dependencies provided by this plugin.

For more information see Gradle documentation on dependency management [https://docs.gradle.org/current/userguide/dependency_management.html#sec:maven_repo].

Note: Google's Maven repo and the Maven Central repo are already included by default.

Note: Only supported on Android and requires EditorExportPlatformAndroid.gradle_build/use_gradle_build to be enabled.

PackedStringArray **_get_android_libraries**(platform: EditorExportPlatform, debug: bool) virtual const 🔗

Virtual method to be overridden by the user. This is called to retrieve the local paths of the Android libraries archive (AAR) files provided by this plugin.

Note: Relative paths **must** be relative to Godot's `res://addons/` directory. For example, an AAR file located under `res://addons/hello_world_plugin/HelloWorld.release.aar` can be returned as an absolute path using `res://addons/hello_world_plugin/HelloWorld.release.aar` or a relative path using `hello_world_plugin/HelloWorld.release.aar`.

Note: Only supported on Android and requires EditorExportPlatformAndroid.gradle_build/use_gradle_build to be enabled.

String **_get_android_manifest_activity_element_contents**(platform: EditorExportPlatform, debug: bool) virtual const

Virtual method to be overridden by the user. This is used at export time to update the contents of the `activity` element in the generated Android manifest.

Note: Only supported on Android and requires EditorExportPlatformAndroid.gradle_build/use_gradle_build to be enabled.

String **_get_android_manifest_application_element_contents**(platform: EditorExportPlatform, debug: bool) virtual const

Virtual method to be overridden by the user. This is used at export time to update the contents of the `application` element in the generated Android manifest.

Note: Only supported on Android and requires EditorExportPlatformAndroid.gradle_build/use_gradle_build to be enabled.

String **_get_android_manifest_element_contents**(platform: EditorExportPlatform, debug: bool) virtual const

Virtual method to be overridden by the user. This is used at export time to update the contents of the `manifest` element in the generated Android manifest.

Note: Only supported on Android and requires EditorExportPlatformAndroid.gradle_build/use_gradle_build to be enabled.

int **_get_customization_configuration_hash**() virtual const

Return a hash based on the configuration passed (for both scenes and resources). This helps keep separate caches for separate export configurations.

Implementing this method is required if _begin_customize_resources returns `true`.

PackedStringArray **_get_export_features**(platform: EditorExportPlatform, debug: bool) virtual const

Return a PackedStringArray of additional features this preset, for the given `platform`, should have.

String **_get_export_option_warning**(platform: EditorExportPlatform, option: String) virtual const

Check the requirements for the given `option` and return a non-empty warning string if they are not met.

Note: Use get_option to check the value of the export options.

Array[Dictionary] **_get_export_options**(platform: EditorExportPlatform) virtual const

Return a list of export options that can be configured for this export plugin.

Each element in the return value is a Dictionary with the following keys:

- `option`: A dictionary with the structure documented by Object.get_property_list, but all keys are optional.
- `default_value`: The default value for this option.
- `update_visibility`: An optional boolean value. If set to `true`, the preset will emit Object.property_list_changed when the option is changed.

Dictionary **_get_export_options_overrides**(platform: EditorExportPlatform) virtual const

Return a Dictionary of override values for export options, that will be used instead of user-provided values. Overridden options will be hidden from the user interface.

```
class MyExportPlugin extends EditorExportPlugin:
    func _get_name() -> String:
        return "MyExportPlugin"

    func _supports_platform(platform) -> bool:
        if platform is EditorExportPlatformPC:
            # Run on all desktop platforms including Windows, MacOS and
Linux.
            return true
        return false

    func _get_export_options_overrides(platform) -> Dictionary:
        # Override "Embed PCK" to always be enabled.
        return {
            "binary_format/embed_pck": true,
        }
```

[String](#) **_get_name**() virtual const [🔗](#)

Return the name identifier of this plugin (for future identification by the exporter). The plugins are sorted by name before exporting.

Implementing this method is required.

[bool](#) **_should_update_export_options**(platform: [EditorExportPlatform](#)) virtual const [🔗](#)

Return `true`, if the result of [_get_export_options](#) has changed and the export options of preset corresponding to `platform` should be updated.

[bool](#) **_supports_platform**(platform: [EditorExportPlatform](#)) virtual const [🔗](#)

Return `true` if the plugin supports the given `platform`.

---

void **add_file**(path: [String](#), file: [PackedByteArray](#), remap: [bool](#)) [🔗](#)

Adds a custom file to be exported. `path` is the virtual path that can be used to load the file, `file` is the binary data of the file.

When called inside [_export_file](#) and `remap` is `true`, the current file will not be exported, but instead remapped to this custom file. `remap` is ignored when called in other places.

`file` will not be imported, so consider using [_customize_resource](#) to remap imported resources.

---

void **add_ios_bundle_file**(path: [String](#)) [🔗](#)

Adds an iOS bundle file from the given `path` to the exported project.

---

void **add_ios_cpp_code**(code: [String](#)) [🔗](#)

Adds a C++ code to the iOS export. The final code is created from the code appended by each active export plugin.

void **add_ios_embedded_framework**(path: String) 🔗

Adds a dynamic library (*.dylib, *.framework) to Linking Phase in iOS's Xcode project and embeds it into resulting binary.

Note: For static libraries (*.a) works in same way as add_ios_framework.

Note: This method should not be used for System libraries as they are already present on the device.

void **add_ios_framework**(path: String) 🔗

Adds a static library (*.a) or dynamic library (*.dylib, *.framework) to Linking Phase in iOS's Xcode project.

void **add_ios_linker_flags**(flags: String) 🔗

Adds linker flags for the iOS export.

void **add_ios_plist_content**(plist_content: String) 🔗

Adds content for iOS Property List files.

void **add_ios_project_static_lib**(path: String) 🔗

Adds a static lib from the given `path` to the iOS project.

void **add_macos_plugin_file**(path: String) 🔗

Adds file or directory matching `path` to `PlugIns` directory of macOS app bundle.

Note: This is useful only for macOS exports.

void **add_shared_object**(path: String, tags: PackedStringArray, target: String) 🔗

Adds a shared object or a directory containing only shared objects with the given `tags` and destination `path`.

Note: In case of macOS exports, those shared objects will be added to `Frameworks` directory of app bundle.

In case of a directory code-sign will error if you place non code object in directory.

Variant **get_option**(name: StringName) const

Returns the current value of an export option supplied by _get_export_options.

---

void **skip**()

To be called inside _export_file. Skips the current file, so it's not included in the export.

EditorFeatureProfile

Inherits: RefCounted **<** Object

An editor feature profile which can be used to disable specific features.

Description

An editor feature profile can be used to disable specific features of the Godot editor. When disabled, the features won't appear in the editor, which makes the editor less cluttered. This is useful in education settings to reduce confusion or when working in a team. For example, artists and level designers could use a feature profile that disables the script editor to avoid accidentally making changes to files they aren't supposed to edit.

To manage editor feature profiles visually, use **Editor > Manage Feature Profiles...** at the top of the editor window.

Methods

String	get_feature_name(feature: Feature)
bool	is_class_disabled(class_name: StringName) const
bool	is_class_editor_disabled(class_name: StringName) const

bool	is_class_property_disabled(class_name: StringName, property: StringName) const
bool	is_feature_disabled(feature: Feature) const
Error	load_from_file(path: String)
Error	save_to_file(path: String)
void	set_disable_class(class_name: StringName, disable: bool)
void	set_disable_class_editor(class_name: StringName, disable: bool)
void	set_disable_class_property(class_name: StringName, property: StringName, disable: bool)
void	set_disable_feature(feature: Feature, disable: bool)

Enumerations

enum **Feature**: 🔗

Feature **FEATURE_3D** = 0

The 3D editor. If this feature is disabled, the 3D editor won't display but 3D nodes will still display in the Create New Node dialog.

Feature **FEATURE_SCRIPT** = `1`

The Script tab, which contains the script editor and class reference browser. If this feature is disabled, the Script tab won't display.

Feature **FEATURE_ASSET_LIB** = `2`

The AssetLib tab. If this feature is disabled, the AssetLib tab won't display.

Feature **FEATURE_SCENE_TREE** = `3`

Scene tree editing. If this feature is disabled, the Scene tree dock will still be visible but will be read-only.

Feature **FEATURE_NODE_DOCK** = `4`

The Node dock. If this feature is disabled, signals and groups won't be visible and modifiable from the editor.

Feature **FEATURE_FILESYSTEM_DOCK** = `5`

The FileSystem dock. If this feature is disabled, the FileSystem dock won't be visible.

Feature **FEATURE_IMPORT_DOCK** = `6`

The Import dock. If this feature is disabled, the Import dock won't be visible.

Feature **FEATURE_HISTORY_DOCK** = `7`

The History dock. If this feature is disabled, the History dock won't be visible.

Feature **FEATURE_MAX** = `8`

Represents the size of the Feature enum.

Method Descriptions

String **get_feature_name**(feature: Feature)

Returns the specified `feature`'s human-readable name.

---

bool **is_class_disabled**(class_name: StringName) const

Returns `true` if the class specified by `class_name` is disabled. When disabled, the class won't appear in the Create New Node dialog.

---

bool **is_class_editor_disabled**(class_name: StringName) const

Returns `true` if editing for the class specified by `class_name` is disabled. When disabled, the class will still appear in the Create New Node dialog but the Inspector will be read-only when selecting a node that extends the class.

---

bool **is_class_property_disabled**(class_name: StringName, property: StringName) const

Returns `true` if `property` is disabled in the class specified by `class_name`. When a property is disabled, it won't appear in the Inspector when selecting a node that extends the class specified by `class_name`.

---

bool **is_feature_disabled**(feature: Feature) const

Returns `true` if the `feature` is disabled. When a feature is disabled, it will disappear from the editor entirely.

Error **load_from_file**(path: String)

Loads an editor feature profile from a file. The file must follow the JSON format obtained by using the feature profile manager's **Export** button or the save_to_file method.

Note: Feature profiles created via the user interface are loaded from the `feature_profiles` directory, as a file with the `.profile` extension. The editor configuration folder can be found by using EditorPaths.get_config_dir.

Error **save_to_file**(path: String)

Saves the editor feature profile to a file in JSON format. It can then be imported using the feature profile manager's **Import** button or the load_from_file method.

Note: Feature profiles created via the user interface are saved in the `feature_profiles` directory, as a file with the `.profile` extension. The editor configuration folder can be found by using EditorPaths.get_config_dir.

---

void **set_disable_class**(class_name: StringName, disable: bool)

If `disable` is `true`, disables the class specified by `class_name`. When disabled, the class won't appear in the Create New Node dialog.

---

void **set_disable_class_editor**(class_name: StringName, disable: bool)

If `disable` is `true`, disables editing for the class specified by `class_name`. When disabled, the class will still appear in the Create New Node dialog but the Inspector will be read-only when selecting a node that extends the class.

---

void **set_disable_class_property**(class_name: StringName, property: StringName, disable: bool) 🔗

If `disable` is `true`, disables editing for `property` in the class specified by `class_name`. When a property is disabled, it won't appear in the Inspector when selecting a node that extends the class specified by `class_name`.

---

void **set_disable_feature**(feature: Feature, disable: bool) 🔗

If `disable` is `true`, disables the editor feature specified in `feature`. When a feature is disabled, it will disappear from the editor entirely.

EditorFileSystemDirectory

Inherits: Object

A directory for the resource filesystem.

Description

A more generalized, low-level variation of the directory concept.

Methods

int	find_dir_index(name: String) const
int	find_file_index(name: String) const
String	get_file(idx: int) const
int	get_file_count() const
bool	get_file_import_is_valid(idx: int) const
String	get_file_path(idx: int) const
String	get_file_script_class_extends(idx: int) const

String	get_file_script_class_name(idx: int) const
StringName	get_file_type(idx: int) const
String	get_name()
EditorFileSystemDirectory	get_parent()
String	get_path() const
EditorFileSystemDirectory	get_subdir(idx: int)
int	get_subdir_count() const

Method Descriptions

int **find_dir_index**(name: String) const 🔗

Returns the index of the directory with name `name` or `-1` if not found.

int **find_file_index**(name: String) const 🔗

Returns the index of the file with name `name` or `-1` if not found.

String **get_file**(idx: int) const 🔗

Returns the name of the file at index `idx`.

---

int **get_file_count**() const 🔗

Returns the number of files in this directory.

---

bool **get_file_import_is_valid**(idx: int) const 🔗

Returns `true` if the file at index `idx` imported properly.

String **get_file_path**(idx: int) const 🔗

Returns the path to the file at index `idx`.

String **get_file_script_class_extends**(idx: int) const 🔗

Returns the base class of the script class defined in the file at index `idx`. If the file doesn't define a script class using the `class_name` syntax, this will return an empty string.

String **get_file_script_class_name**(idx: int) const 🔗

Returns the name of the script class defined in the file at index `idx`. If the file doesn't define a script class using the `class_name` syntax, this will return an empty string.

StringName **get_file_type**(idx: int) const 🔗

Returns the resource type of the file at index `idx`. This returns a string such as `"Resource"` or `"GDScript"`, *not* a file extension such as `".gd"`.

String **get_name**()

Returns the name of this directory.

EditorFileSystemDirectory **get_parent**()

Returns the parent directory for this directory or `null` if called on a directory at `res://` or `user://`.

String **get_path**() const

Returns the path to this directory.

EditorFileSystemDirectory **get_subdir**(idx: int)

Returns the subdirectory at index `idx`.

---

int **get_subdir_count**() const

Returns the number of subdirectories in this directory.

EditorFileSystemImportFormatSupportQuery

Inherits: RefCounted **<** Object

Used to query and configure import format support.

Description

This class is used to query and configure a certain import format. It is used in conjunction with asset format import plugins.

Methods

PackedStringArray	_get_file_extensions() virtual const
bool	_is_active() virtual const
bool	_query() virtual const

Method Descriptions

PackedStringArray **_get_file_extensions**() virtual const 🔗

Return the file extensions supported.

---

bool **_is_active**() virtual const 🔗

Return whether this importer is active.

---

bool **_query**() virtual const 🔗

Query support. Return false if import must not continue.

EditorImportPlugin

Inherits: ResourceImporter **<** RefCounted **<** Object

Registers a custom resource importer in the editor. Use the class to parse any file and import it as a new resource type.

Description

EditorImportPlugins provide a way to extend the editor's resource import functionality. Use them to import resources from custom files or to provide alternatives to the editor's existing importers.

EditorImportPlugins work by associating with specific file extensions and a resource type. See _get_recognized_extensions and _get_resource_type. They may optionally specify some import presets that affect the import process. EditorImportPlugins are responsible for creating the resources and saving them in the `.godot/imported` directory (see ProjectSettings.application/config/use_hidden_project_data_directory).

Below is an example EditorImportPlugin that imports a Mesh from a file with the extension ".special" or ".spec":

GDScript

```
@tool
extends EditorImportPlugin

func _get_importer_name():
    return "my.special.plugin"

func _get_visible_name():
    return "Special Mesh"

func _get_recognized_extensions():
    return ["special", "spec"]

func _get_save_extension():
    return "mesh"

func _get_resource_type():
    return "Mesh"

func _get_preset_count():
    return 1

func _get_preset_name(preset_index):
    return "Default"

func _get_import_options(path, preset_index):
    return [{"name": "my_option", "default_value": false}]

func _import(source_file, save_path, options,
platform_variants, gen_files):
    var file = FileAccess.open(source_file, FileAccess.READ)
    if file == null:
        return FAILED
    var mesh = ArrayMesh.new()
    # Fill the Mesh with data read in "file", left as an
exercise to the reader.

    var filename = save_path + "." + _get_save_extension()
    return ResourceSaver.save(mesh, filename)
```

C#

```
using Godot;

public partial class MySpecialPlugin : EditorImportPlugin
{
    public override string _GetImporterName()
    {
        return "my.special.plugin";
    }
    public override string _GetVisibleName()
    {
        return "Special Mesh";
    }

    public override string[] _GetRecognizedExtensions()
    {
        return new string[] { "special", "spec" };
    }

    public override string _GetSaveExtension()
    {
        return "mesh";
    }

    public override string _GetResourceType()
    {
        return "Mesh";
    }

    public override int _GetPresetCount()
    {
        return 1;
    }

    public override string _GetPresetName(int presetIndex)
    {
        return "Default";
    }

    public override
Godot.Collections.Array<Godot.Collections.Dictionary>
_GetImportOptions(string path, int presetIndex)
    {
        return new
Godot.Collections.Array<Godot.Collections.Dictionary>
        {
            new Godot.Collections.Dictionary
            {
                { "name", "myOption" },
                { "default_value", false },
            }
        };
    }

    public override Error _Import(string sourceFile, string
savePath, Godot.Collections.Dictionary options,
Godot.Collections.Array<string> platformVariants,
Godot.Collections.Array<string> genFiles)
    {
        using var file = FileAccess.Open(sourceFile,
FileAccess.ModeFlags.Read);
        if (file.GetError() != Error.Ok)
        {
            return Error.Failed;
        }

        var mesh = new ArrayMesh();
        // Fill the Mesh with data read in "file", left as an
exercise to the reader.
        string filename = $"{savePath}.{_GetSaveExtension()}";
        return ResourceSaver.Save(mesh, filename);
    }
}
```

To use **EditorImportPlugin**, register it using the EditorPlugin.add_import_plugin method first.

Tutorials

- Import plugins

Methods

bool	_can_import_threaded() virtual const

Array[Dictionary]	_get_import_options(path: String, preset_index: int) virtual const
int	_get_import_order() virtual const
String	_get_importer_name() virtual const
bool	_get_option_visibility(path: String, option_name: StringName, options: Dictionary) virtual const
int	_get_preset_count() virtual const
String	_get_preset_name(preset_index: int) virtual const
float	_get_priority() virtual const
PackedStringArray	_get_recognized_extensions() virtual const
String	_get_resource_type() virtual const
String	_get_save_extension() virtual const
String	_get_visible_name() virtual const

Error	_import(source_file: String, save_path: String, options: Dictionary, platform_variants: Array[String], gen_files: Array[String]) virtual const
Error	append_import_external_resource(path: String, custom_options: Dictionary = {}, custom_importer: String = "", generator_parameters: Variant = null)

Method Descriptions

bool **_can_import_threaded**() virtual const 🔗

Tells whether this importer can be run in parallel on threads, or, on the contrary, it's only safe for the editor to call it from the main thread, for one file at a time.

If this method is not overridden, it will return `true` by default (i.e., safe for parallel importing).

Array[Dictionary] **_get_import_options**(path: String, preset_index: int) virtual const 🔗

Gets the options and default values for the preset at this index. Returns an Array of Dictionaries with the following keys: `name`, `default_value`, `property_hint` (optional), `hint_string` (optional), `usage` (optional).

---

int **_get_import_order**() virtual const 🔗

Gets the order of this importer to be run when importing resources. Importers with *lower* import orders will be called first, and higher values will be called later. Use this to ensure the importer runs after the dependencies are already imported. The default import order is `0` unless overridden by a specific importer. See ImportOrder for some predefined values.

String **_get_importer_name**() virtual const 🔗

Gets the unique name of the importer.

---

bool **_get_option_visibility**(path: String, option_name: StringName, options: Dictionary) virtual const 🔗

This method can be overridden to hide specific import options if conditions are met. This is mainly useful for hiding options that depend on others if one of them is disabled. For example:

GDScript

```
func _get_option_visibility(option, options):
    # Only show the lossy quality setting if the compression
mode is set to "Lossy".
    if option == "compress/lossy_quality" and
options.has("compress/mode"):
        return int(options["compress/mode"]) == COMPRESS_LOSSY
# This is a constant that you set

    return true
```

C#

```
public void _GetOptionVisibility(string option,
Godot.Collections.Dictionary options)
{
    // Only show the lossy quality setting if the compression
mode is set to "Lossy".
```

```
    if (option == "compress/lossy_quality" &&
options.ContainsKey("compress/mode"))
    {
        return (int)options["compress/mode"] == CompressLossy;
// This is a constant you set
    }

    return true;
}
```

Returns `true` to make all options always visible.

---

int **_get_preset_count**() virtual const 🔗

Gets the number of initial presets defined by the plugin. Use _get_import_options to get the default options for the preset and _get_preset_name to get the name of the preset.

String **_get_preset_name**(preset_index: int) virtual const 🔗

Gets the name of the options preset at this index.

---

float **_get_priority**() virtual const 🔗

Gets the priority of this plugin for the recognized extension. Higher priority plugins will be preferred. The default priority is `1.0`.

PackedStringArray **_get_recognized_extensions**() virtual const 🔗

Gets the list of file extensions to associate with this loader (case-insensitive). e.g. `["obj"]`.

String **_get_resource_type**() virtual const

Gets the Godot resource type associated with this loader. e.g. `"Mesh"` or `"Animation"`.

String **_get_save_extension**() virtual const

Gets the extension used to save this resource in the `.godot/imported` directory (see ProjectSettings.application/config/use_hidden_project_data_directory).

String **_get_visible_name**() virtual const

Gets the name to display in the import window. You should choose this name as a continuation to "Import as", e.g. "Import as Special Mesh".

Error **_import**(source_file: String, save_path: String, options: Dictionary, platform_variants: Array[String], gen_files: Array[String]) virtual const

Imports `source_file` into `save_path` with the import `options` specified. The `platform_variants` and `gen_files` arrays will be modified by this function.

This method must be overridden to do the actual importing work. See this class' description for an example of overriding this method.

Error **append_import_external_resource**(path: String, custom_options: Dictionary = {}, custom_importer: String = "", generator_parameters: Variant = null)

This function can only be called during the _import callback and it allows manually importing resources from it. This is useful when the imported file generates external resources that require importing (as example, images). Custom parameters for the ".import" file can be passed via the `custom_options`. Additionally, in cases where multiple importers can handle a file, the `custom_importer` can be specified to force a specific one. This function performs a resource import and returns immediately with a success or error code. `generator_parameters` defines optional extra metadata which will be stored as `generator_parameters` in the `remap` section of the `.import` file, for example to store a md5 hash of the source data.

EditorInspectorPlugin

Inherits: RefCounted **<** Object

Plugin for adding custom property editors on the inspector.

Description

EditorInspectorPlugin allows adding custom property editors to EditorInspector.

When an object is edited, the _can_handle function is called and must return `true` if the object type is supported.

If supported, the function _parse_begin will be called, allowing to place custom controls at the beginning of the class.

Subsequently, the _parse_category and _parse_property are called for every category and property. They offer the ability to add custom controls to the inspector too.

Finally, _parse_end will be called.

On each of these calls, the "add" functions can be called.

To use **EditorInspectorPlugin**, register it using the EditorPlugin.add_inspector_plugin method first.

Tutorials

- Inspector plugins

Methods

bool	_can_handle(object: Object) virtual const
void	_parse_begin(object: Object) virtual
void	_parse_category(object: Object, category: String) virtual
void	_parse_end(object: Object) virtual
void	_parse_group(object: Object, group: String) virtual
bool	_parse_property(object: Object, type: Variant.Type, name: String, hint_type: PropertyHint, hint_string: String, usage_flags: BitField[PropertyUsageFlags], wide: bool) virtual
void	add_custom_control(control: Control)
void	add_property_editor(property: String, editor: Control, add_to_end: bool = false, label: String = "")
void	add_property_editor_for_multiple_properties(label: String, properties: PackedStringArray, editor: Control)

Method Descriptions

bool **_can_handle**(object: Object) virtual const

Returns `true` if this object can be handled by this plugin.

void **_parse_begin**(object: Object) virtual

Called to allow adding controls at the beginning of the property list for `object`.

void **_parse_category**(object: Object, category: String) virtual

Called to allow adding controls at the beginning of a category in the property list for `object`.

void **_parse_end**(object: Object) virtual

Called to allow adding controls at the end of the property list for `object`.

void **_parse_group**(object: Object, group: String) virtual

Called to allow adding controls at the beginning of a group or a sub-group in the property list for `object`.

bool **_parse_property**(object: Object, type: Variant.Type, name: String, hint_type: PropertyHint, hint_string: String, usage_flags: BitField[PropertyUsageFlags], wide: bool) virtual

Called to allow adding property-specific editors to the property list for `object`. The added editor control must extend EditorProperty.

Returning `true` removes the built-in editor for this property, otherwise allows to insert a custom editor before the built-in one.

---

void **add_custom_control**(control: Control)

Adds a custom control, which is not necessarily a property editor.

---

void **add_property_editor**(property: String, editor: Control, add_to_end: bool = false, label: String = "")

Adds a property editor for an individual property. The `editor` control must extend EditorProperty.

There can be multiple property editors for a property. If `add_to_end` is `true`, this newly added editor will be displayed after all the other editors of the property whose `add_to_end` is `false`. For example, the editor uses this parameter to add an "Edit Region" button for Sprite2D.region_rect below the regular Rect2 editor.

`label` can be used to choose a custom label for the property editor in the inspector. If left empty, the label is computed from the name of the property instead.

---

void **add_property_editor_for_multiple_properties**(label: String, properties: PackedStringArray, editor: Control)

Adds an editor that allows modifying multiple properties. The `editor` control must extend EditorProperty.

EditorInterface

Inherits: Object

Godot editor's interface.

Description

EditorInterface gives you control over Godot editor's window. It allows customizing the window, saving and (re-)loading scenes, rendering mesh previews, inspecting and editing resources and objects, and provides access to EditorSettings, EditorFileSystem, EditorResourcePreview, ScriptEditor, the editor viewport, and information about scenes.

Note: This class shouldn't be instantiated directly. Instead, access the singleton directly by its name.

GDScript

```
var editor_settings = EditorInterface.get_editor_settings()
```

C#

```
// In C# you can access it via the static Singleton property.
EditorSettings settings =
EditorInterface.Singleton.GetEditorSettings();
```

Properties

bool	distraction_free_mode

bool	movie_maker_enabled

Methods

void	edit_node(node: Node)
void	edit_resource(resource: Resource)
void	edit_script(script: Script, line: int = -1, column: int = 0, grab_focus: bool = true)
Control	get_base_control() const
EditorCommandPalette	get_command_palette() const
String	get_current_directory() const
String	get_current_feature_profile() const
String	get_current_path() const
Node	get_edited_scene_root() const
VBoxContainer	get_editor_main_screen() const

EditorPaths	get_editor_paths() const
float	get_editor_scale() const
EditorSettings	get_editor_settings() const
Theme	get_editor_theme() const
SubViewport	get_editor_viewport_2d() const
SubViewport	get_editor_viewport_3d(idx: int = 0) const
FileSystemDock	get_file_system_dock() const
EditorInspector	get_inspector() const
PackedStringArray	get_open_scenes() const
String	get_playing_scene() const
EditorFileSystem	get_resource_filesystem() const
EditorResourcePreview	get_resource_previewer() const
ScriptEditor	get_script_editor() const

PackedStringArray	get_selected_paths() const
EditorSelection	get_selection() const
void	inspect_object(object: Object, for_property: String = "", inspector_only: bool = false)
bool	is_multi_window_enabled() const
bool	is_playing_scene() const
bool	is_plugin_enabled(plugin: String) const
Array[Texture2D]	make_mesh_previews(meshes: Array[Mesh], preview_size: int)
void	mark_scene_as_unsaved()
void	open_scene_from_path(scene_filepath: String)
void	play_current_scene()
void	play_custom_scene(scene_filepath: String)

void	play_main_scene()
void	popup_dialog(dialog: Window, rect: Rect2i = Rect2i(0, 0, 0, 0))
void	popup_dialog_centered(dialog: Window, minsize: Vector2i = Vector2i(0, 0))
void	popup_dialog_centered_clamped(dialog: Window, minsize: Vector2i = Vector2i(0, 0), fallback_ratio: float = 0.75)
void	popup_dialog_centered_ratio(dialog: Window, ratio: float = 0.8)
void	popup_node_selector(callback: Callable, valid_types: Array[StringName] = [])
void	popup_property_selector(object: Object, callback: Callable, type_filter: PackedInt32Array = PackedInt32Array())
void	reload_scene_from_path(scene_filepath: String)
void	restart_editor(save: bool = true)

void	save_all_scenes()
Error	save_scene()
void	save_scene_as(path: String, with_preview: bool = true)
void	select_file(file: String)
void	set_current_feature_profile(profile_name: String)
void	set_main_screen_editor(name: String)
void	set_plugin_enabled(plugin: String, enabled: bool)
void	stop_playing_scene()

Property Descriptions

bool **distraction_free_mode** 🔗

- void **set_distraction_free_mode**(value: bool)
- bool **is_distraction_free_mode_enabled**()

If `true`, enables distraction-free mode which hides side docks to increase the space available for the main view.

---

bool **movie_maker_enabled**

- void **set_movie_maker_enabled**(value: bool)
- bool **is_movie_maker_enabled**()

If `true`, the Movie Maker mode is enabled in the editor. See MovieWriter for more information.

Method Descriptions

void **edit_node**(node: Node)

Edits the given Node. The node will be also selected if it's inside the scene tree.

---

void **edit_resource**(resource: Resource)

Edits the given Resource. If the resource is a Script you can also edit it with edit_script to specify the line and column position.

---

void **edit_script**(script: Script, line: int = -1, column: int = 0, grab_focus: bool = true)

Edits the given Script. The line and column on which to open the script can also be specified. The script will be open with the user-configured editor for the script's language which may be an external editor.

Control **get_base_control**() const

Returns the main container of Godot editor's window. For example, you can use it to retrieve the size of the container and place your controls accordingly.

Warning: Removing and freeing this node will render the editor useless and may cause a crash.

EditorCommandPalette **get_command_palette**() const

Returns the editor's EditorCommandPalette instance.

Warning: Removing and freeing this node will render a part of the editor useless and may cause a crash.

String **get_current_directory**() const

Returns the current directory being viewed in the FileSystemDock. If a file is selected, its base directory will be returned using String.get_base_dir instead.

String **get_current_feature_profile**() const

Returns the name of the currently activated feature profile. If the default profile is currently active, an empty string is returned instead.

In order to get a reference to the EditorFeatureProfile, you must load the feature profile using EditorFeatureProfile.load_from_file.

Note: Feature profiles created via the user interface are loaded from the `feature_profiles` directory, as a file with the `.profile` extension.

The editor configuration folder can be found by using EditorPaths.get_config_dir.

String **get_current_path**() const 🔗

Returns the current path being viewed in the FileSystemDock.

Node **get_edited_scene_root**() const 🔗

Returns the edited (current) scene's root Node.

VBoxContainer **get_editor_main_screen**() const 🔗

Returns the editor control responsible for main screen plugins and tools. Use it with plugins that implement EditorPlugin._has_main_screen.

Note: This node is a VBoxContainer, which means that if you add a Control child to it, you need to set the child's Control.size_flags_vertical to Control.SIZE_EXPAND_FILL to make it use the full available space.

Warning: Removing and freeing this node will render a part of the editor useless and may cause a crash.

EditorPaths **get_editor_paths**() const 🔗

Returns the EditorPaths singleton.

---

float **get_editor_scale**() const 🔗

Returns the actual scale of the editor UI (`1.0` being 100% scale). This can be used to adjust position and dimensions of the UI added by plugins.

Note: This value is set via the `interface/editor/display_scale` and `interface/editor/custom_display_scale` editor settings. Editor must be restarted for changes to be properly applied.

EditorSettings **get_editor_settings**() const 🔗

Returns the editor's EditorSettings instance.

Theme **get_editor_theme**() const 🔗

Returns the editor's Theme.

Note: When creating custom editor UI, prefer accessing theme items directly from your GUI nodes using the `get_theme_*` methods.

SubViewport **get_editor_viewport_2d**() const 🔗

Returns the 2D editor SubViewport. It does not have a camera. Instead, the view transforms are done directly and can be accessed with Viewport.global_canvas_transform.

SubViewport **get_editor_viewport_3d**(idx: int = 0) const 🔗

Returns the specified 3D editor SubViewport, from `0` to `3`. The viewport can be used to access the active editor cameras with Viewport.get_camera_3d.

FileSystemDock **get_file_system_dock**() const 🔗

Returns the editor's FileSystemDock instance.

Warning: Removing and freeing this node will render a part of the editor useless and may cause a crash.

EditorInspector **get_inspector**() const 🔗

Returns the editor's EditorInspector instance.

Warning: Removing and freeing this node will render a part of the editor useless and may cause a crash.

PackedStringArray **get_open_scenes**() const 🔗

Returns an Array with the file paths of the currently opened scenes.

String **get_playing_scene**() const 🔗

Returns the name of the scene that is being played. If no scene is currently being played, returns an empty string.

EditorFileSystem **get_resource_filesystem**() const 🔗

Returns the editor's EditorFileSystem instance.

EditorResourcePreview **get_resource_previewer**() const 🔗

Returns the editor's EditorResourcePreview instance.

ScriptEditor **get_script_editor**() const 🔗

Returns the editor's ScriptEditor instance.

Warning: Removing and freeing this node will render a part of the editor useless and may cause a crash.

PackedStringArray **get_selected_paths**() const 🔗

Returns an array containing the paths of the currently selected files (and directories) in the FileSystemDock.

EditorSelection **get_selection**() const 🔗

Returns the editor's EditorSelection instance.

---

void **inspect_object**(object: Object, for_property: String = "", inspector_only: bool = false) 🔗

Shows the given property on the given `object` in the editor's Inspector dock. If `inspector_only` is `true`, plugins will not attempt to edit `object`.

---

bool **is_multi_window_enabled**() const 🔗

Returns `true` if multiple window support is enabled in the editor. Multiple window support is enabled if *all* of these statements are true:

- EditorSettings.interface/multi_window/enable is `true`.
- EditorSettings.interface/editor/single_window_mode is `false`.
- Viewport.gui_embed_subwindows is `false`. This is forced to `true` on platforms that don't support multiple windows such as

Web, or when the `--single-window` command line argument is used.

bool **is_playing_scene**() const

Returns `true` if a scene is currently being played, `false` otherwise. Paused scenes are considered as being played.

bool **is_plugin_enabled**(plugin: String) const

Returns `true` if the specified `plugin` is enabled. The plugin name is the same as its directory name.

Array[Texture2D] **make_mesh_previews**(meshes: Array[Mesh], preview_size: int)

Returns mesh previews rendered at the given size as an Array of Texture2Ds.

void **mark_scene_as_unsaved**()

Marks the current scene tab as unsaved.

void **open_scene_from_path**(scene_filepath: String)

Opens the scene at the given path.

void **play_current_scene**()

Plays the currently active scene.

void **play_custom_scene**(scene_filepath: String)

Plays the scene specified by its filepath.

---

void **play_main_scene**()

Plays the main scene.

---

void **popup_dialog**(dialog: Window, rect: Rect2i = Rect2i(0, 0, 0, 0))

Pops up the `dialog` in the editor UI with Window.popup_exclusive. The dialog must have no current parent, otherwise the method fails.

See also Window.set_unparent_when_invisible.

---

void **popup_dialog_centered**(dialog: Window, minsize: Vector2i = Vector2i(0, 0))

Pops up the `dialog` in the editor UI with Window.popup_exclusive_centered. The dialog must have no current parent, otherwise the method fails.

See also Window.set_unparent_when_invisible.

---

void **popup_dialog_centered_clamped**(dialog: Window, minsize: Vector2i = Vector2i(0, 0), fallback_ratio: float = 0.75)

Pops up the `dialog` in the editor UI with Window.popup_exclusive_centered_clamped. The dialog must have no current parent, otherwise the method fails.

See also Window.set_unparent_when_invisible.

---

void **popup_dialog_centered_ratio**(dialog: Window, ratio: float = 0.8)

Pops up the `dialog` in the editor UI with Window.popup_exclusive_centered_ratio. The dialog must have no current parent, otherwise the method fails.

See also Window.set_unparent_when_invisible.

---

void **popup_node_selector**(callback: Callable, valid_types: Array[StringName] = [])

Pops up an editor dialog for selecting a Node from the edited scene. The `callback` must take a single argument of type NodePath. It is called on the selected NodePath or the empty path `^""` if the dialog is canceled. If `valid_types` is provided, the dialog will only show Nodes that match one of the listed Node types.

Example:

```
func _ready():
    if Engine.is_editor_hint():
        EditorInterface.popup_node_selector(_on_node_selected, ["Button"])

func _on_node_selected(node_path):
    if node_path.is_empty():
        print("node selection canceled")
    else:
        print("selected ", node_path)
```

---

void **popup_property_selector**(object: Object, callback: Callable, type_filter: PackedInt32Array = PackedInt32Array())

Pops up an editor dialog for selecting properties from `object`. The `callback` must take a single argument of type NodePath. It is called on the selected property path (see NodePath.get_as_property_path) or the empty path `^""` if the dialog is canceled. If `type_filter` is provided, the dialog will only show properties that match one of the listed Variant.Type values.

Example:

```
func _ready():
    if Engine.is_editor_hint():
        EditorInterface.popup_property_selector(this, _on_property_selected, [TYPE_INT])

func _on_property_selected(property_path):
    if property_path.is_empty():
        print("property selection canceled")
    else:
        print("selected ", property_path)
```

---

void **reload_scene_from_path**(scene_filepath: String)

Reloads the scene at the given path.

---

void **restart_editor**(save: bool = true)

Restarts the editor. This closes the editor and then opens the same project. If `save` is `true`, the project will be saved before restarting.

---

void **save_all_scenes**()

Saves all opened scenes in the editor.

Error **save_scene**()

Saves the currently active scene. Returns either @GlobalScope.OK or @GlobalScope.ERR_CANT_CREATE.

void **save_scene_as**(path: String, with_preview: bool = true)

Saves the currently active scene as a file at `path`.

void **select_file**(file: String)

Selects the file, with the path provided by `file`, in the FileSystem dock.

void **set_current_feature_profile**(profile_name: String)

Selects and activates the specified feature profile with the given `profile_name`. Set `profile_name` to an empty string to reset to the default feature profile.

A feature profile can be created programmatically using the EditorFeatureProfile class.

Note: The feature profile that gets activated must be located in the `feature_profiles` directory, as a file with the `.profile` extension. If a profile could not be found, an error occurs. The editor configuration folder can be found by using EditorPaths.get_config_dir.

void **set_main_screen_editor**(name: String)

Sets the editor's current main screen to the one specified in `name`. `name` must match the title of the tab in question exactly (e.g. `2D`, `3D`, `Script`, or `AssetLib` for default tabs).

void **set_plugin_enabled**(plugin: String, enabled: bool) 🔗

Sets the enabled status of a plugin. The plugin name is the same as its directory name.

void **stop_playing_scene**() 🔗

Stops the scene that is currently playing.

EditorNode3DGizmo

Inherits: Node3DGizmo **<** RefCounted **<** Object

Gizmo for editing Node3D objects.

Description

Gizmo that is used for providing custom visualization and editing (handles and subgizmos) for Node3D objects. Can be overridden to create custom gizmos, but for simple gizmos creating a EditorNode3DGizmoPlugin is usually recommended.

Methods

void	_begin_handle_action(id: int, secondary: bool) virtual
void	_commit_handle(id: int, secondary: bool, restore: Variant, cancel: bool) virtual
void	_commit_subgizmos(ids: PackedInt32Array, restores: Array[Transform3D], cancel: bool) virtual

String	_get_handle_name(id: int, secondary: bool) virtual const
Variant	_get_handle_value(id: int, secondary: bool) virtual const
Transform3D	_get_subgizmo_transform(id: int) virtual const
bool	_is_handle_highlighted(id: int, secondary: bool) virtual const
void	_redraw() virtual
void	_set_handle(id: int, secondary: bool, camera: Camera3D, point: Vector2) virtual
void	_set_subgizmo_transform(id: int, transform: Transform3D) virtual
PackedInt32Array	_subgizmos_intersect_frustum(camera: Camera3D, frustum: Array[Plane]) virtual const
int	_subgizmos_intersect_ray(camera: Camera3D, point: Vector2) virtual const

void	add_collision_segments(segments: PackedVector3Array)
void	add_collision_triangles(triangles: TriangleMesh)
void	add_handles(handles: PackedVector3Array, material: Material, ids: PackedInt32Array, billboard: bool = false, secondary: bool = false)
void	add_lines(lines: PackedVector3Array, material: Material, billboard: bool = false, modulate: Color = Color(1, 1, 1, 1))
void	add_mesh(mesh: Mesh, material: Material = null, transform: Transform3D = Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0), skeleton: SkinReference = null)
void	add_unscaled_billboard(material: Material, default_scale: float = 1, modulate: Color = Color(1, 1, 1, 1))
void	clear()
Node3D	get_node_3d() const

EditorNode3DGizmoPlugin	get_plugin() const
PackedInt32Array	get_subgizmo_selection() const
bool	is_subgizmo_selected(id: int) const
void	set_hidden(hidden: bool)
void	set_node_3d(node: Node)

Method Descriptions

void **_begin_handle_action**(id: int, secondary: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_commit_handle**(id: int, secondary: bool, restore: Variant, cancel: bool) virtual 🔗

Override this method to commit a handle being edited (handles must have been previously added by add_handles). This usually means creating an UndoRedo action for the change, using the current handle value as "do" and the `restore` argument as "undo".

If the `cancel` argument is `true`, the `restore` value should be directly set, without any UndoRedo action.

The `secondary` argument is `true` when the committed handle is secondary (see add_handles for more information).

---

void **_commit_subgizmos**(ids: PackedInt32Array, restores: Array[Transform3D], cancel: bool) virtual 🔗

Override this method to commit a group of subgizmos being edited (see _subgizmos_intersect_ray and _subgizmos_intersect_frustum). This usually means creating an UndoRedo action for the change, using the current transforms as "do" and the `restores` transforms as "undo".

If the `cancel` argument is `true`, the `restores` transforms should be directly set, without any UndoRedo action.

String **_get_handle_name**(id: int, secondary: bool) virtual const 🔗

Override this method to return the name of an edited handle (handles must have been previously added by add_handles). Handles can be named for reference to the user when editing.

The `secondary` argument is `true` when the requested handle is secondary (see add_handles for more information).

Variant **_get_handle_value**(id: int, secondary: bool) virtual const 🔗

Override this method to return the current value of a handle. This value will be requested at the start of an edit and used as the `restore` argument in _commit_handle.

The `secondary` argument is `true` when the requested handle is secondary (see add_handles for more information).

Transform3D **_get_subgizmo_transform**(id: int) virtual const 🔗

Override this method to return the current transform of a subgizmo. This transform will be requested at the start of an edit and used as the `restore` argument in _commit_subgizmos.

---

bool **_is_handle_highlighted**(id: int, secondary: bool) virtual const 🔗

Override this method to return `true` whenever the given handle should be highlighted in the editor.

The `secondary` argument is `true` when the requested handle is secondary (see add_handles for more information).

---

void **_redraw**() virtual 🔗

Override this method to add all the gizmo elements whenever a gizmo update is requested. It's common to call clear at the beginning of this method and then add visual elements depending on the node's properties.

---

void **_set_handle**(id: int, secondary: bool, camera: Camera3D, point: Vector2) virtual 🔗

Override this method to update the node properties when the user drags a gizmo handle (previously added with add_handles). The provided `point` is the mouse position in screen coordinates and the `camera` can be used to convert it to raycasts.

The `secondary` argument is `true` when the edited handle is secondary (see add_handles for more information).

---

void **_set_subgizmo_transform**(id: int, transform: Transform3D) virtual 🔗

Override this method to update the node properties during subgizmo editing (see _subgizmos_intersect_ray and _subgizmos_intersect_frustum). The `transform` is given in the Node3D's local coordinate system.

PackedInt32Array **_subgizmos_intersect_frustum**(camera: Camera3D, frustum: Array[Plane]) virtual const 🔗

Override this method to allow selecting subgizmos using mouse drag box selection. Given a `camera` and a `frustum`, this method should return which subgizmos are contained within the frustum. The `frustum` argument consists of an array with all the Planes that make up the selection frustum. The returned value should contain a list of unique subgizmo identifiers, which can have any non-negative value and will be used in other virtual methods like _get_subgizmo_transform or _commit_subgizmos.

---

int **_subgizmos_intersect_ray**(camera: Camera3D, point: Vector2) virtual const 🔗

Override this method to allow selecting subgizmos using mouse clicks. Given a `camera` and a `point` in screen coordinates, this method should return which subgizmo should be selected. The returned value should be a unique subgizmo identifier, which can have any non-negative value and will be used in other virtual methods like _get_subgizmo_transform or _commit_subgizmos.

---

void **add_collision_segments**(segments: PackedVector3Array) 🔗

Adds the specified `segments` to the gizmo's collision shape for picking. Call this method during _redraw.

---

void **add_collision_triangles**(triangles: TriangleMesh) 🔗

Adds collision triangles to the gizmo for picking. A TriangleMesh can be generated from a regular Mesh too. Call this method during _redraw.

---

void **add_handles**(handles: PackedVector3Array, material: Material, ids: PackedInt32Array, billboard: bool = false, secondary: bool = false) 🔗

Adds a list of handles (points) which can be used to edit the properties of the gizmo's Node3D. The `ids` argument can be used to specify a custom identifier for each handle, if an empty array is passed, the ids will be assigned automatically from the `handles` argument order.

The `secondary` argument marks the added handles as secondary, meaning they will normally have lower selection priority than regular handles. When the user is holding the shift key secondary handles will switch to have higher priority than regular handles. This change in priority can be used to place multiple handles at the same point while still giving the user control on their selection.

There are virtual methods which will be called upon editing of these handles. Call this method during _redraw.

---

void **add_lines**(lines: PackedVector3Array, material: Material, billboard: bool = false, modulate: Color = Color(1, 1, 1, 1)) 🔗

Adds lines to the gizmo (as sets of 2 points), with a given material. The lines are used for visualizing the gizmo. Call this method during _redraw.

---

void **add_mesh**(mesh: Mesh, material: Material = null, transform: Transform3D = Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0), skeleton: SkinReference = null)

Adds a mesh to the gizmo with the specified `material`, local `transform` and `skeleton`. Call this method during _redraw.

---

void **add_unscaled_billboard**(material: Material, default_scale: float = 1, modulate: Color = Color(1, 1, 1, 1))

Adds an unscaled billboard for visualization and selection. Call this method during _redraw.

---

void **clear**()

Removes everything in the gizmo including meshes, collisions and handles.

Node3D **get_node_3d**() const

Returns the Node3D node associated with this gizmo.

EditorNode3DGizmoPlugin **get_plugin**() const

Returns the EditorNode3DGizmoPlugin that owns this gizmo. It's useful to retrieve materials using EditorNode3DGizmoPlugin.get_material.

PackedInt32Array **get_subgizmo_selection**() const 🔗

Returns a list of the currently selected subgizmos. Can be used to highlight selected elements during _redraw.

---

bool **is_subgizmo_selected**(id: int) const 🔗

Returns `true` if the given subgizmo is currently selected. Can be used to highlight selected elements during _redraw.

---

void **set_hidden**(hidden: bool) 🔗

Sets the gizmo's hidden state. If `true`, the gizmo will be hidden. If `false`, it will be shown.

---

void **set_node_3d**(node: Node) 🔗

Sets the reference Node3D node for the gizmo. `node` must inherit from Node3D.

EditorPaths

Inherits: Object

Editor-only singleton that returns paths to various OS-specific data folders and files.

Description

This editor-only singleton returns OS-specific paths to various data folders and files. It can be used in editor plugins to ensure files are saved in the correct location on each operating system.

Note: This singleton is not accessible in exported projects. Attempting to access it in an exported project will result in a script error as the singleton won't be declared. To prevent script errors in exported projects, use Engine.has_singleton to check whether the singleton is available before using it.

Note: On the Linux/BSD platform, Godot complies with the XDG Base Directory Specification [https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html]. You can override environment variables following the specification to change the editor and project data paths.

Tutorials

- File paths in Godot projects

Methods

String	get_cache_dir() const
String	get_config_dir() const
String	get_data_dir() const
String	get_project_settings_dir() const
String	get_self_contained_file() const
bool	is_self_contained() const

Method Descriptions

String **get_cache_dir**() const

Returns the absolute path to the user's cache folder. This folder should be used for temporary data that can be removed safely whenever the editor is closed (such as generated resource thumbnails).

Default paths per platform:

```
- Windows: %LOCALAPPDATA%\Godot\
- macOS: ~/Library/Caches/Godot/
- Linux: ~/.cache/godot/
```

String **get_config_dir**() const

Returns the absolute path to the user's configuration folder. This folder should be used for *persistent* user configuration files.

Default paths per platform:

```
- Windows: %APPDATA%\Godot\                    (same as
`get_data_dir()`)
- macOS: ~/Library/Application Support/Godot/  (same as
`get_data_dir()`)
- Linux: ~/.config/godot/
```

String **get_data_dir**() const

Returns the absolute path to the user's data folder. This folder should be used for *persistent* user data files such as installed export templates.

Default paths per platform:

```
- Windows: %APPDATA%\Godot\                    (same as
`get_config_dir()`)
- macOS: ~/Library/Application Support/Godot/  (same as
`get_config_dir()`)
- Linux: ~/.local/share/godot/
```

String **get_project_settings_dir**() const

Returns the project-specific editor settings path. Projects all have a unique subdirectory inside the settings path where project-specific editor settings are saved.

String **get_self_contained_file**() const

Returns the absolute path to the self-contained file that makes the current Godot editor instance be considered as self-contained.

Returns an empty string if the current Godot editor instance isn't self-contained. See also is_self_contained.

---

bool **is_self_contained**() const 🔗

Returns `true` if the editor is marked as self-contained, `false` otherwise. When self-contained mode is enabled, user configuration, data and cache files are saved in an `editor_data/` folder next to the editor binary. This makes portable usage easier and ensures the Godot editor minimizes file writes outside its own folder. Self-contained mode is not available for exported projects.

Self-contained mode can be enabled by creating a file named `._sc_` or `_sc_` in the same folder as the editor binary or macOS .app bundle while the editor is not running. See also get_self_contained_file.

Note: On macOS, quarantine flag should be manually removed before using self-contained mode, see Running on macOS [https://docs.godotengine.org/en/stable/tutorials/export/running_on_macos.html].

Note: On macOS, placing `_sc_` or any other file inside .app bundle will break digital signature and make it non-portable, consider placing it in the same folder as the .app bundle instead.

Note: The Steam release of Godot uses self-contained mode by default.

EditorResourceConversionPlugin

Inherits: RefCounted **<** Object

Plugin for adding custom converters from one resource format to another in the editor resource picker context menu; for example, converting a StandardMaterial3D to a ShaderMaterial.

Description

EditorResourceConversionPlugin is invoked when the context menu is brought up for a resource in the editor inspector. Relevant conversion plugins will appear as menu options to convert the given resource to a target type.

Below shows an example of a basic plugin that will convert an ImageTexture to a PortableCompressedTexture2D.

GDScript

```
extends EditorResourceConversionPlugin

func _handles(resource: Resource):
    return resource is ImageTexture

func _converts_to():
    return "PortableCompressedTexture2D"

func _convert(itex: Resource):
    var ptex = PortableCompressedTexture2D.new()
    ptex.create_from_image(itex.get_image(),
PortableCompressedTexture2D.COMPRESSION_MODE_LOSSLESS)
    return ptex
```

To use an **EditorResourceConversionPlugin**, register it using the EditorPlugin.add_resource_conversion_plugin method first.

Methods

Resource	_convert(resource: Resource) virtual const
String	_converts_to() virtual const
bool	_handles(resource: Resource) virtual const

Method Descriptions

Resource **_convert**(resource: Resource) virtual const 🔗

Takes an input Resource and converts it to the type given in _converts_to. The returned Resource is the result of the conversion, and the input Resource remains unchanged.

String **_converts_to**() virtual const 🔗

Returns the class name of the target type of Resource that this plugin converts source resources to.

---

bool **_handles**(resource: Resource) virtual const 🔗

Called to determine whether a particular Resource can be converted to the target resource type by this plugin.

EditorResourcePreviewGenerator

Inherits: RefCounted **<** Object

Custom generator of previews.

Description

Custom code to generate previews. Please check `file_dialog/thumbnail_size` in EditorSettings to find out the right size to do previews at.

Methods

bool	_can_generate_small_preview() virtual const
Texture2D	_generate(resource: Resource, size: Vector2i, metadata: Dictionary) virtual const
Texture2D	_generate_from_path(path: String, size: Vector2i, metadata: Dictionary) virtual const
bool	_generate_small_preview_automatically() virtual const

bool	_handles(type: String) virtual const

Method Descriptions

bool **_can_generate_small_preview**() virtual const

If this function returns `true`, the generator will call _generate or _generate_from_path for small previews as well.

By default, it returns `false`.

Texture2D **_generate**(resource: Resource, size: Vector2i, metadata: Dictionary) virtual const

Generate a preview from a given resource with the specified size. This must always be implemented.

Returning an empty texture is an OK way to fail and let another generator take care.

Care must be taken because this function is always called from a thread (not the main thread).

`metadata` dictionary can be modified to store file-specific metadata that can be used in EditorResourceTooltipPlugin._make_tooltip_for_path (like image size, sample length etc.).

Texture2D **_generate_from_path**(path: String, size: Vector2i, metadata: Dictionary) virtual const

Generate a preview directly from a path with the specified size. Implementing this is optional, as default code will load and call _generate.

Returning an empty texture is an OK way to fail and let another generator take care.

Care must be taken because this function is always called from a thread (not the main thread).

`metadata` dictionary can be modified to store file-specific metadata that can be used in EditorResourceTooltipPlugin._make_tooltip_for_path (like image size, sample length etc.).

---

bool **_generate_small_preview_automatically**() virtual const 🔗

If this function returns `true`, the generator will automatically generate the small previews from the normal preview texture generated by the methods _generate or _generate_from_path.

By default, it returns `false`.

---

bool **_handles**(type: String) virtual const 🔗

Returns `true` if your generator supports the resource of type `type`.

EditorResourceTooltipPlugin

Inherits: RefCounted **<** Object

A plugin that advanced tooltip for its handled resource type.

Description

Resource tooltip plugins are used by FileSystemDock to generate customized tooltips for specific resources. E.g. tooltip for a Texture2D displays a bigger preview and the texture's dimensions.

A plugin must be first registered with FileSystemDock.add_resource_tooltip_plugin. When the user hovers a resource in filesystem dock which is handled by the plugin, _make_tooltip_for_path is called to create the tooltip. It works similarly to Control._make_custom_tooltip.

Methods

bool	_handles(type: String) virtual const
Control	_make_tooltip_for_path(path: String, metadata: Dictionary, base: Control) virtual const
void	request_thumbnail(path: String, control: TextureRect) const

Method Descriptions

bool **_handles**(type: String) virtual const

Return `true` if the plugin is going to handle the given Resource `type`.

Control **_make_tooltip_for_path**(path: String, metadata: Dictionary, base: Control) virtual const

Create and return a tooltip that will be displayed when the user hovers a resource under the given `path` in filesystem dock.

The `metadata` dictionary is provided by preview generator (see EditorResourcePreviewGenerator._generate).

`base` is the base default tooltip, which is a VBoxContainer with a file name, type and size labels. If another plugin handled the same file type, `base` will be output from the previous plugin. For best result, make sure the base tooltip is part of the returned Control.

Note: It's unadvised to use ResourceLoader.load, especially with heavy resources like models or textures, because it will make the editor unresponsive when creating the tooltip. You can use request_thumbnail if you want to display a preview in your tooltip.

Note: If you decide to discard the `base`, make sure to call Node.queue_free, because it's not freed automatically.

```
func _make_tooltip_for_path(path, metadata, base):
    var t_rect = TextureRect.new()
    request_thumbnail(path, t_rect)
    base.add_child(t_rect) # The TextureRect will appear at the bottom of the tooltip.
    return base
```

---

void **request_thumbnail**(path: String, control: TextureRect) const

Requests a thumbnail for the given TextureRect. The thumbnail is created asynchronously by EditorResourcePreview and automatically set when available.

EditorSceneFormatImporter

Inherits: RefCounted **<** Object

Inherited By: EditorSceneFormatImporterBlend, EditorSceneFormatImporterFBX2GLTF, EditorSceneFormatImporterGLTF, EditorSceneFormatImporterUFBX

Imports scenes from third-parties' 3D files.

Description

EditorSceneFormatImporter allows to define an importer script for a third-party 3D format.

To use **EditorSceneFormatImporter**, register it using the EditorPlugin.add_scene_format_importer_plugin method first.

Methods

PackedStringArray	_get_extensions() virtual const
int	_get_import_flags() virtual const
void	_get_import_options(path: String) virtual
Variant	_get_option_visibility(path: String, for_animation: bool, option: String) virtual const

Object	_import_scene(path: String, flags: int, options: Dictionary) virtual

Constants

IMPORT_SCENE = `1` 🔗

There is currently no description for this constant. Please help us by contributing one!

IMPORT_ANIMATION = `2` 🔗

There is currently no description for this constant. Please help us by contributing one!

IMPORT_FAIL_ON_MISSING_DEPENDENCIES = `4` 🔗

There is currently no description for this constant. Please help us by contributing one!

IMPORT_GENERATE_TANGENT_ARRAYS = `8` 🔗

There is currently no description for this constant. Please help us by contributing one!

IMPORT_USE_NAMED_SKIN_BINDS = `16` 🔗

There is currently no description for this constant. Please help us by contributing one!

IMPORT_DISCARD_MESHES_AND_MATERIALS = `32` 🔗

There is currently no description for this constant. Please help us by contributing one!

IMPORT_FORCE_DISABLE_MESH_COMPRESSION = `64`

There is currently no description for this constant. Please help us by contributing one!

Method Descriptions

PackedStringArray **_get_extensions**() virtual const

There is currently no description for this method. Please help us by contributing one!

---

int **_get_import_flags**() virtual const

There is currently no description for this method. Please help us by contributing one!

---

void **_get_import_options**(path: String) virtual

There is currently no description for this method. Please help us by contributing one!

Variant **_get_option_visibility**(path: String, for_animation: bool, option: String) virtual const

There is currently no description for this method. Please help us by contributing one!

Object **_import_scene**(path: String, flags: int, options: Dictionary) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

EditorSceneFormatImporterBlend

Inherits: EditorSceneFormatImporter **<** RefCounted **<** Object

Importer for Blender's `.blend` scene file format.

Description

Imports Blender scenes in the `.blend` file format through the glTF 2.0 3D import pipeline. This importer requires Blender to be installed by the user, so that it can be used to export the scene as glTF 2.0.

The location of the Blender binary is set via the `filesystem/import/blender/blender_path` editor setting.

This importer is only used if ProjectSettings.filesystem/import/blender/enabled is enabled, otherwise `.blend` files present in the project folder are not imported.

Blend import requires Blender 3.0.

Internally, the EditorSceneFormatImporterBlend uses the Blender glTF "Use Original" mode to reference external textures.

EditorSceneFormatImporterFBX2GLTF

Inherits: EditorSceneFormatImporter **<** RefCounted **<** Object

Importer for the `.fbx` scene file format.

Description

Imports Autodesk FBX 3D scenes by way of converting them to glTF 2.0 using the FBX2glTF command line tool.

The location of the FBX2glTF binary is set via the EditorSettings.filesystem/import/fbx/fbx2gltf_path editor setting.

This importer is only used if ProjectSettings.filesystem/import/fbx2gltf/enabled is set to `true`.

EditorSceneFormatImporterGLTF

Inherits: EditorSceneFormatImporter **<** RefCounted **<** Object

There is currently no description for this class. Please help us by contributing one!

EditorSceneFormatImporterUFBX

Experimental: This class may be changed or removed in future versions.

Inherits: EditorSceneFormatImporter **<** RefCounted **<** Object

Import FBX files using the ufbx library.

Description

EditorSceneFormatImporterUFBX is designed to load FBX files and supports both binary and ASCII FBX files from version 3000 onward. This class supports various 3D object types like meshes, skins, blend shapes, materials, and rigging information. The class aims for feature parity with the official FBX SDK and supports FBX 7.4 specifications.

EditorScenePostImport

Inherits: RefCounted **<** Object

Post-processes scenes after import.

Description

Imported scenes can be automatically modified right after import by setting their **Custom Script** Import property to a `tool` script that inherits from this class.

The _post_import callback receives the imported scene's root node and returns the modified version of the scene. Usage example:

GDScript

```
@tool # Needed so it runs in editor.
extends EditorScenePostImport

# This sample changes all node names.
# Called right after the scene is imported and gets the root node.
func _post_import(scene):
    # Change all node names to "modified_[oldnodename]"
    iterate(scene)
    return scene # Remember to return the imported scene

func iterate(node):
    if node != null:
        node.name = "modified_" + node.name
        for child in node.get_children():
            iterate(child)
```

C#

```
using Godot;

// This sample changes all node names.
// Called right after the scene is imported and gets the root node.
[Tool]
public partial class NodeRenamer : EditorScenePostImport
{
    public override GodotObject _PostImport(Node scene)
    {
        // Change all node names to "modified_[oldnodename]"
        Iterate(scene);
        return scene; // Remember to return the imported scene
    }

    public void Iterate(Node node)
    {
        if (node != null)
        {
            node.Name = $"modified_{node.Name}";
            foreach (Node child in node.GetChildren())
            {
                Iterate(child);
            }
        }
    }
}
```

Tutorials

- Importing 3D scenes: Configuration: Using import scripts for automation

Methods

Object _post_import(scene: Node) virtual

String	get_source_file() const

Method Descriptions

Object **_post_import**(scene: Node) virtual 🔗

Called after the scene was imported. This method must return the modified version of the scene.

String **get_source_file**() const 🔗

Returns the source file path which got imported (e.g. `res://scene.dae`).

EditorScenePostImportPlugin

Inherits: RefCounted **<** Object

Plugin to control and modifying the process of importing a scene.

Description

This plugin type exists to modify the process of importing scenes, allowing to change the content as well as add importer options at every stage of the process.

Methods

void	_get_import_options(path: String) virtual
void	_get_internal_import_options(category: int) virtual
Variant	_get_internal_option_update_view_required(category: int, option: String) virtual const
Variant	_get_internal_option_visibility(category: int, for_animation: bool, option: String) virtual const
Variant	_get_option_visibility(path: String, for_animation: bool, option: String) virtual const

void	_internal_process(category: int, base_node: Node, node: Node, resource: Resource) virtual
void	_post_process(scene: Node) virtual
void	_pre_process(scene: Node) virtual
void	add_import_option(name: String, value: Variant)
void	add_import_option_advanced(type: Variant.Type, name: String, default_value: Variant, hint: PropertyHint = 0, hint_string: String = "", usage_flags: int = 6)
Variant	get_option_value(name: StringName) const

Enumerations

enum **InternalImportCategory**: 🔗

InternalImportCategory **INTERNAL_IMPORT_CATEGORY_NODE** = 0

There is currently no description for this enum. Please help us by contributing one!

InternalImportCategory **INTERNAL_IMPORT_CATEGORY_MESH_3D_NODE** = 1

There is currently no description for this enum. Please help us by contributing one!

InternalImportCategory **INTERNAL_IMPORT_CATEGORY_MESH** = `2`

There is currently no description for this enum. Please help us by contributing one!

InternalImportCategory **INTERNAL_IMPORT_CATEGORY_MATERIAL** = `3`

There is currently no description for this enum. Please help us by contributing one!

InternalImportCategory **INTERNAL_IMPORT_CATEGORY_ANIMATION** = `4`

There is currently no description for this enum. Please help us by contributing one!

InternalImportCategory **INTERNAL_IMPORT_CATEGORY_ANIMATION_NODE** = `5`

There is currently no description for this enum. Please help us by contributing one!

InternalImportCategory **INTERNAL_IMPORT_CATEGORY_SKELETON_3D_NODE** = `6`

There is currently no description for this enum. Please help us by contributing one!

InternalImportCategory **INTERNAL_IMPORT_CATEGORY_MAX** = `7`

There is currently no description for this enum. Please help us by contributing one!

Method Descriptions

void **_get_import_options**(path: String) virtual

Override to add general import options. These will appear in the main import dock on the editor. Add options via add_import_option and add_import_option_advanced.

void **_get_internal_import_options**(category: int) virtual

Override to add internal import options. These will appear in the 3D scene import dialog. Add options via add_import_option and add_import_option_advanced.

Variant **_get_internal_option_update_view_required**(category: int, option: String) virtual const

Return true whether updating the 3D view of the import dialog needs to be updated if an option has changed.

Variant **_get_internal_option_visibility**(category: int, for_animation: bool, option: String) virtual const

Return true or false whether a given option should be visible. Return null to ignore.

Variant **_get_option_visibility**(path: String, for_animation: bool, option: String) virtual const 🔗

Return true or false whether a given option should be visible. Return null to ignore.

void **_internal_process**(category: int, base_node: Node, node: Node, resource: Resource) virtual 🔗

Process a specific node or resource for a given category.

void **_post_process**(scene: Node) virtual 🔗

Post process the scene. This function is called after the final scene has been configured.

void **_pre_process**(scene: Node) virtual 🔗

Pre Process the scene. This function is called right after the scene format loader loaded the scene and no changes have been made.

void **add_import_option**(name: String, value: Variant) 🔗

Add a specific import option (name and default value only). This function can only be called from _get_import_options and _get_internal_import_options.

void **add_import_option_advanced**(type: Variant.Type, name: String, default_value: Variant, hint: PropertyHint = 0, hint_string: String = "", usage_flags: int = 6) 🔗

Add a specific import option. This function can only be called from _get_import_options and _get_internal_import_options.

Variant **get_option_value**(name: StringName) const 🔗

Query the value of an option. This function can only be called from those querying visibility, or processing.

EditorScript

Inherits: RefCounted < Object

Base script that can be used to add extension functions to the editor.

Description

Scripts extending this class and implementing its _run method can be executed from the Script Editor's **File > Run** menu option (or by pressing `Ctrl + Shift + X`) while the editor is running. This is useful for adding custom in-editor functionality to Godot. For more complex additions, consider using EditorPlugins instead.

Note: Extending scripts need to have `tool` mode enabled.

Example script:

GDScript

```
@tool
extends EditorScript

func _run():
    print("Hello from the Godot Editor!")
```

C#

```
using Godot;

[Tool]
public partial class HelloEditor : EditorScript
{
    public override void _Run()
    {
        GD.Print("Hello from the Godot Editor!");
    }
}
```

Note: The script is run in the Editor context, which means the output is visible in the console window started with the Editor (stdout) instead of the usual Godot **Output** dock.

Note: EditorScript is RefCounted, meaning it is destroyed when nothing references it. This can cause errors during asynchronous operations if there are no references to the script.

Methods

void	_run() virtual
void	add_root_node(node: Node)
EditorInterface	get_editor_interface() const
Node	get_scene() const

Method Descriptions

void **_run**() virtual 🔗

This method is executed by the Editor when **File > Run** is used.

---

void **add_root_node**(node: Node) 🔗

Makes `node` root of the currently opened scene. Only works if the scene is empty. If the `node` is a scene instance, an inheriting scene will be created.

EditorInterface **get_editor_interface**() const 🔗

Deprecated: EditorInterface is a global singleton and can be accessed directly by its name.

Returns the EditorInterface singleton instance.

Node **get_scene**() const 🔗

Returns the edited (current) scene's root Node. Equivalent of EditorInterface.get_edited_scene_root.

EditorSelection

Inherits: Object

Manages the SceneTree selection in the editor.

Description

This object manages the SceneTree selection in the editor.

Note: This class shouldn't be instantiated directly. Instead, access the singleton using EditorInterface.get_selection.

Methods

void	add_node(node: Node)
void	clear()
Array[Node]	get_selected_nodes()
Array[Node]	get_transformable_selected_nodes()
void	remove_node(node: Node)

Signals

selection_changed()

Emitted when the selection changes.

Method Descriptions

void **add_node**(node: Node)

Adds a node to the selection.

Note: The newly selected node will not be automatically edited in the inspector. If you want to edit a node, use EditorInterface.edit_node.

void **clear**()

Clear the selection.

Array[Node] **get_selected_nodes**()

Returns the list of selected nodes.

Array[Node] **get_transformable_selected_nodes**()

Returns the list of selected nodes, optimized for transform operations (i.e. moving them, rotating, etc.). This list can be used to avoid situations where a node is selected and is also a child/grandchild.

void **remove_node**(node: Node) 🔗

Removes a node from the selection.

EditorTranslationParserPlugin

Inherits: RefCounted < Object

Plugin for adding custom parsers to extract strings that are to be translated from custom files (.csv, .json etc.).

Description

EditorTranslationParserPlugin is invoked when a file is being parsed to extract strings that require translation. To define the parsing and string extraction logic, override the _parse_file method in script.

Add the extracted strings to argument `msgids` or `msgids_context_plural` if context or plural is used.

When adding to `msgids_context_plural`, you must add the data using the format `["A", "B", "C"]`, where `A` represents the extracted string, `B` represents the context, and `C` represents the plural version of the extracted string. If you want to add only context but not plural, put `""` for the plural slot. The idea is the same if you only want to add plural but not context. See the code below for concrete examples.

The extracted strings will be written into a POT file selected by user under "POT Generation" in "Localization" tab in "Project Settings" menu.

Below shows an example of a custom parser that extracts strings from a CSV file to write into a POT.

GDScript

```
@tool
extends EditorTranslationParserPlugin

func _parse_file(path, msgids, msgids_context_plural):
    var file = FileAccess.open(path, FileAccess.READ)
    var text = file.get_as_text()
    var split_strs = text.split(",", false)
    for s in split_strs:
        msgids.append(s)
        #print("Extracted string: " + s)

func _get_recognized_extensions():
    return ["csv"]
```

C#

```
using Godot;

[Tool]
public partial class CustomParser :
EditorTranslationParserPlugin
{
    public override void _ParseFile(string path,
Godot.Collections.Array<string> msgids,
Godot.Collections.Array<Godot.Collections.Array>
msgidsContextPlural)
    {
        using var file = FileAccess.Open(path,
FileAccess.ModeFlags.Read);
        string text = file.GetAsText();
        string[] splitStrs = text.Split(",", allowEmpty:
false);
        foreach (string s in splitStrs)
        {
            msgids.Add(s);
            //GD.Print($"Extracted string: {s}");
        }
    }

    public override string[] _GetRecognizedExtensions()
    {
        return new string[] { "csv" };
    }
}
```

To add a translatable string associated with context or plural, add it to `msgids_context_plural`:

GDScript

```
# This will add a message with msgid "Test 1", msgctxt "context", and msgid_plural "test 1 plurals".
msgids_context_plural.append(["Test 1", "context", "test 1 plurals"])
# This will add a message with msgid "A test without context" and msgid_plural "plurals".
msgids_context_plural.append(["A test without context", "", "plurals"])
# This will add a message with msgid "Only with context" and msgctxt "a friendly context".
msgids_context_plural.append(["Only with context", "a friendly context", ""])
```

C#

```
// This will add a message with msgid "Test 1", msgctxt "context", and msgid_plural "test 1 plurals".
msgidsContextPlural.Add(new Godot.Collections.Array{"Test 1", "context", "test 1 Plurals"});
// This will add a message with msgid "A test without context" and msgid_plural "plurals".
msgidsContextPlural.Add(new Godot.Collections.Array{"A test without context", "", "plurals"});
// This will add a message with msgid "Only with context" and msgctxt "a friendly context".
msgidsContextPlural.Add(new Godot.Collections.Array{"Only with context", "a friendly context", ""});
```

Note: If you override parsing logic for standard script types (GDScript, C#, etc.), it would be better to load the `path` argument using ResourceLoader.load. This is because built-in scripts are loaded as Resource type, not FileAccess type.

For example:

GDScript

```
func _parse_file(path, msgids, msgids_context_plural):
    var res = ResourceLoader.load(path, "Script")
    var text = res.source_code
    # Parsing logic.

func _get_recognized_extensions():
    return ["gd"]
```

C#

```
public override void _ParseFile(string path,
Godot.Collections.Array<string> msgids,
Godot.Collections.Array<Godot.Collections.Array>
msgidsContextPlural)
{
    var res = ResourceLoader.Load<Script>(path, "Script");
    string text = res.SourceCode;
    // Parsing logic.
}

public override string[] _GetRecognizedExtensions()
{
    return new string[] { "gd" };
}
```

To use **EditorTranslationParserPlugin**, register it using the EditorPlugin.add_translation_parser_plugin method first.

Methods

PackedStringArray _get_recognized_extensions() virtual const

---

void	_parse_file(path: String, msgids: Array[String], msgids_context_plural: Array[Array]) virtual

Method Descriptions

PackedStringArray **_get_recognized_extensions**() virtual const 🔗

Gets the list of file extensions to associate with this parser, e.g. `["csv"]`.

void **_parse_file**(path: String, msgids: Array[String], msgids_context_plural: Array[Array]) virtual 🔗

Override this method to define a custom parsing logic to extract the translatable strings.

EditorUndoRedoManager

Inherits: Object

Manages undo history of scenes opened in the editor.

Description

EditorUndoRedoManager is a manager for UndoRedo objects associated with edited scenes. Each scene has its own undo history and **EditorUndoRedoManager** ensures that each action performed in the editor gets associated with a proper scene. For actions not related to scenes (ProjectSettings edits, external resources, etc.), a separate global history is used.

The usage is mostly the same as UndoRedo. You create and commit actions and the manager automatically decides under-the-hood what scenes it belongs to. The scene is deduced based on the first operation in an action, using the object from the operation. The rules are as follows:

- If the object is a Node, use the currently edited scene;
- If the object is a built-in resource, use the scene from its path;
- If the object is external resource or anything else, use global history.

This guessing can sometimes yield false results, so you can provide a custom context object when creating an action.

EditorUndoRedoManager is intended to be used by Godot editor plugins. You can obtain it using EditorPlugin.get_undo_redo. For non-editor uses or plugins that don't need to integrate with the editor's undo history, use UndoRedo instead.

The manager's API is mostly the same as in UndoRedo, so you can refer to its documentation for more examples. The main difference is that **EditorUndoRedoManager** uses object + method name for actions, instead of Callable.

Methods

void	add_do_method(object: Object, method: StringName, ...) vararg
void	add_do_property(object: Object, property: StringName, value: Variant)
void	add_do_reference(object: Object)
void	add_undo_method(object: Object, method: StringName, ...) vararg
void	add_undo_property(object: Object, property: StringName, value: Variant)
void	add_undo_reference(object: Object)
void	commit_action(execute: bool = true)
void	create_action(name: String, merge_mode: MergeMode = 0, custom_context: Object = null, backward_undo_ops: bool = false)

void	force_fixed_history()
UndoRedo	get_history_undo_redo(id: int) const
int	get_object_history_id(object: Object) const
bool	is_committing_action() const

Signals

history_changed()

Emitted when the list of actions in any history has changed, either when an action is committed or a history is cleared.

version_changed()

Emitted when the version of any history has changed as a result of undo or redo call.

Enumerations

enum **SpecialHistory**:

SpecialHistory **GLOBAL_HISTORY** = 0

Global history not associated with any scene, but with external resources etc.

SpecialHistory **REMOTE_HISTORY** = `-9`

History associated with remote inspector. Used when live editing a running project.

SpecialHistory **INVALID_HISTORY** = `-99`

Invalid "null" history. It's a special value, not associated with any object.

Method Descriptions

void **add_do_method**(object: Object, method: StringName, ...) vararg 🔗

Register a method that will be called when the action is committed (i.e. the "do" action).

If this is the first operation, the `object` will be used to deduce target undo history.

---

void **add_do_property**(object: Object, property: StringName, value: Variant) 🔗

Register a property value change for "do".

If this is the first operation, the `object` will be used to deduce target undo history.

---

void **add_do_reference**(object: Object) 🔗

Register a reference for "do" that will be erased if the "do" history is lost. This is useful mostly for new nodes created for the "do" call. Do not use for resources.

---

void **add_undo_method**(object: Object, method: StringName, ...) vararg 🔗

Register a method that will be called when the action is undone (i.e. the "undo" action).

If this is the first operation, the `object` will be used to deduce target undo history.

---

void **add_undo_property**(object: Object, property: StringName, value: Variant) 🔗

Register a property value change for "undo".

If this is the first operation, the `object` will be used to deduce target undo history.

---

void **add_undo_reference**(object: Object) 🔗

Register a reference for "undo" that will be erased if the "undo" history is lost. This is useful mostly for nodes removed with the "do" call (not the "undo" call!).

---

void **commit_action**(execute: bool = true) 🔗

Commit the action. If `execute` is true (default), all "do" methods/properties are called/set when this function is called.

---

void **create_action**(name: String, merge_mode: MergeMode = 0, custom_context: Object = null, backward_undo_ops: bool = false)

Create a new action. After this is called, do all your calls to add_do_method, add_undo_method, add_do_property, and add_undo_property, then commit the action with commit_action.

The way actions are merged is dictated by the `merge_mode` argument. See MergeMode for details.

If `custom_context` object is provided, it will be used for deducing target history (instead of using the first operation).

The way undo operation are ordered in actions is dictated by `backward_undo_ops`. When `backward_undo_ops` is `false` undo option are ordered in the same order they were added. Which means the first operation to be added will be the first to be undone.

void **force_fixed_history**()

Forces the next operation (e.g. add_do_method) to use the action's history rather than guessing it from the object. This is sometimes needed when a history can't be correctly determined, like for a nested resource that doesn't have a path yet.

This method should only be used when absolutely necessary, otherwise it might cause invalid history state. For most of complex cases, the `custom_context` parameter of create_action is sufficient.

UndoRedo **get_history_undo_redo**(id: int) const

Returns the UndoRedo object associated with the given history `id`.

`id` above `0` are mapped to the opened scene tabs (but it doesn't match their order). `id` of `0` or lower have special meaning (see SpecialHistory).

Best used with get_object_history_id. This method is only provided in case you need some more advanced methods of UndoRedo (but keep in mind that directly operating on the UndoRedo object might affect editor's stability).

---

int **get_object_history_id**(object: Object) const

Returns the history ID deduced from the given `object`. It can be used with get_history_undo_redo.

---

bool **is_committing_action**() const

Returns `true` if the **EditorUndoRedoManager** is currently committing the action, i.e. running its "do" method or property change (see commit_action).

EditorVCSInterface

Inherits: Object

Version Control System (VCS) interface, which reads and writes to the local VCS in use.

Description

Defines the API that the editor uses to extract information from the underlying VCS. The implementation of this API is included in VCS plugins, which are GDExtension plugins that inherit **EditorVCSInterface** and are attached (on demand) to the singleton instance of **EditorVCSInterface**. Instead of performing the task themselves, all the virtual functions listed below are calling the internally overridden functions in the VCS plugins to provide a plug-n-play experience. A custom VCS plugin is supposed to inherit from **EditorVCSInterface** and override each of these virtual functions.

Tutorials

- Version control systems

Methods

bool	_checkout_branch(branch_name: String) virtual
void	_commit(msg: String) virtual

void	_create_branch(branch_name: String) virtual
void	_create_remote(remote_name: String, remote_url: String) virtual
void	_discard_file(file_path: String) virtual
void	_fetch(remote: String) virtual
Array[String]	_get_branch_list() virtual
String	_get_current_branch_name() virtual
Array[Dictionary]	_get_diff(identifier: String, area: int) virtual
Array[Dictionary]	_get_line_diff(file_path: String, text: String) virtual
Array[Dictionary]	_get_modified_files_data() virtual
Array[Dictionary]	_get_previous_commits(max_commits: int) virtual
Array[String]	_get_remotes() virtual

String	_get_vcs_name() virtual
bool	_initialize(project_path: String) virtual
void	_pull(remote: String) virtual
void	_push(remote: String, force: bool) virtual
void	_remove_branch(branch_name: String) virtual
void	_remove_remote(remote_name: String) virtual
void	_set_credentials(username: String, password: String, ssh_public_key_path: String, ssh_private_key_path: String, ssh_passphrase: String) virtual
bool	_shut_down() virtual
void	_stage_file(file_path: String) virtual
void	_unstage_file(file_path: String) virtual
Dictionary	add_diff_hunks_into_diff_file(diff_file: Dictionary, diff_hunks: Array[Dictionary])

Dictionary	add_line_diffs_into_diff_hunk(diff_hunk: Dictionary, line_diffs: Array[Dictionary])
Dictionary	create_commit(msg: String, author: String, id: String, unix_timestamp: int, offset_minutes: int)
Dictionary	create_diff_file(new_file: String, old_file: String)
Dictionary	create_diff_hunk(old_start: int, new_start: int, old_lines: int, new_lines: int)
Dictionary	create_diff_line(new_line_no: int, old_line_no: int, content: String, status: String)
Dictionary	create_status_file(file_path: String, change_type: ChangeType, area: TreeArea)
void	popup_error(msg: String)

Enumerations

enum **ChangeType**: 🔗

ChangeType **CHANGE_TYPE_NEW** = `0`

A new file has been added.

ChangeType **CHANGE_TYPE_MODIFIED** = `1`

An earlier added file has been modified.

ChangeType **CHANGE_TYPE_RENAMED** = `2`

An earlier added file has been renamed.

ChangeType **CHANGE_TYPE_DELETED** = `3`

An earlier added file has been deleted.

ChangeType **CHANGE_TYPE_TYPECHANGE** = `4`

An earlier added file has been typechanged.

ChangeType **CHANGE_TYPE_UNMERGED** = `5`

A file is left unmerged.

---

enum **TreeArea**: 🔗

TreeArea **TREE_AREA_COMMIT** = `0`

A commit is encountered from the commit area.

TreeArea **TREE_AREA_STAGED** = `1`

A file is encountered from the staged area.

TreeArea **TREE_AREA_UNSTAGED** = `2`

A file is encountered from the unstaged area.

Method Descriptions

bool **_checkout_branch**(branch_name: String) virtual 🔗

Checks out a `branch_name` in the VCS.

---

void **_commit**(msg: String) virtual

Commits the currently staged changes and applies the commit `msg` to the resulting commit.

---

void **_create_branch**(branch_name: String) virtual

Creates a new branch named `branch_name` in the VCS.

---

void **_create_remote**(remote_name: String, remote_url: String) virtual

Creates a new remote destination with name `remote_name` and points it to `remote_url`. This can be an HTTPS remote or an SSH remote.

---

void **_discard_file**(file_path: String) virtual

Discards the changes made in a file present at `file_path`.

---

void **_fetch**(remote: String) virtual

Fetches new changes from the `remote`, but doesn't write changes to the current working directory. Equivalent to `git fetch`.

Array[String] **_get_branch_list**() virtual

Gets an instance of an Array of Strings containing available branch names in the VCS.

String **_get_current_branch_name**() virtual

Gets the current branch name defined in the VCS.

Array[Dictionary] **_get_diff**(identifier: String, area: int) virtual

Returns an array of Dictionary items (see create_diff_file, create_diff_hunk, create_diff_line, add_line_diffs_into_diff_hunk and add_diff_hunks_into_diff_file), each containing information about a diff. If `identifier` is a file path, returns a file diff, and if it is a commit identifier, then returns a commit diff.

Array[Dictionary] **_get_line_diff**(file_path: String, text: String) virtual

Returns an Array of Dictionary items (see create_diff_hunk), each containing a line diff between a file at `file_path` and the `text` which is passed in.

Array[Dictionary] **_get_modified_files_data**() virtual

Returns an Array of Dictionary items (see create_status_file), each containing the status data of every modified file in the project folder.

Array[Dictionary] **_get_previous_commits**(max_commits: int) virtual

Returns an Array of Dictionary items (see create_commit), each containing the data for a past commit.

Array[String] **_get_remotes**() virtual

Returns an Array of Strings, each containing the name of a remote configured in the VCS.

String **_get_vcs_name**() virtual

Returns the name of the underlying VCS provider.

---

bool **_initialize**(project_path: String) virtual

Initializes the VCS plugin when called from the editor. Returns whether or not the plugin was successfully initialized. A VCS project is initialized at `project_path`.

---

void **_pull**(remote: String) virtual

Pulls changes from the remote. This can give rise to merge conflicts.

---

void **_push**(remote: String, force: bool) virtual

Pushes changes to the `remote`. If `force` is `true`, a force push will override the change history already present on the remote.

---

void **_remove_branch**(branch_name: String) virtual

Remove a branch from the local VCS.

---

void **_remove_remote**(remote_name: String) virtual

Remove a remote from the local VCS.

void **_set_credentials**(username: String, password: String, ssh_public_key_path: String, ssh_private_key_path: String, ssh_passphrase: String) virtual 🔗

Set user credentials in the underlying VCS. `username` and `password` are used only during HTTPS authentication unless not already mentioned in the remote URL. `ssh_public_key_path`, `ssh_private_key_path`, and `ssh_passphrase` are only used during SSH authentication.

bool **_shut_down**() virtual 🔗

Shuts down VCS plugin instance. Called when the user either closes the editor or shuts down the VCS plugin through the editor UI.

void **_stage_file**(file_path: String) virtual 🔗

Stages the file present at `file_path` to the staged area.

void **_unstage_file**(file_path: String) virtual 🔗

Unstages the file present at `file_path` from the staged area to the unstaged area.

Dictionary **add_diff_hunks_into_diff_file**(diff_file: Dictionary, diff_hunks: Array[Dictionary]) 🔗

Helper function to add an array of `diff_hunks` into a `diff_file`.

Dictionary **add_line_diffs_into_diff_hunk**(diff_hunk: Dictionary, line_diffs: Array[Dictionary])

Helper function to add an array of `line_diffs` into a `diff_hunk`.

Dictionary **create_commit**(msg: String, author: String, id: String, unix_timestamp: int, offset_minutes: int)

Helper function to create a commit Dictionary item. `msg` is the commit message of the commit. `author` is a single human-readable string containing all the author's details, e.g. the email and name configured in the VCS. `id` is the identifier of the commit, in whichever format your VCS may provide an identifier to commits. `unix_timestamp` is the UTC Unix timestamp of when the commit was created. `offset_minutes` is the timezone offset in minutes, recorded from the system timezone where the commit was created.

Dictionary **create_diff_file**(new_file: String, old_file: String)

Helper function to create a Dictionary for storing old and new diff file paths.

Dictionary **create_diff_hunk**(old_start: int, new_start: int, old_lines: int, new_lines: int)

Helper function to create a Dictionary for storing diff hunk data. `old_start` is the starting line number in old file. `new_start` is the starting line number in new file. `old_lines` is the number of lines in the old file. `new_lines` is the number of lines in the new file.

Dictionary **create_diff_line**(new_line_no: int, old_line_no: int, content: String, status: String)

Helper function to create a Dictionary for storing a line diff. `new_line_no` is the line number in the new file (can be `-1` if the line is deleted). `old_line_no` is the line number in the old file (can be `-1` if the line is added). `content` is the diff text. `status` is a single character string which stores the line origin.

Dictionary **create_status_file**(file_path: String, change_type: ChangeType, area: TreeArea)

Helper function to create a Dictionary used by editor to read the status of a file.

---

void **popup_error**(msg: String)

Pops up an error message in the editor which is shown as coming from the underlying VCS. Use this to show VCS specific error messages.

EncodedObjectAsID

Inherits: RefCounted **<** Object

Holds a reference to an Object's instance ID.

Description

Utility class which holds a reference to the internal identifier of an Object instance, as given by Object.get_instance_id. This ID can then be used to retrieve the object instance with @GlobalScope.instance_from_id.

This class is used internally by the editor inspector and script debugger, but can also be used in plugins to pass and display objects as their IDs.

Properties

int	object_id	0

Property Descriptions

int **object_id** = 0 🔗

- void **set_object_id**(value: int)
- int **get_object_id**()

The Object identifier stored in this **EncodedObjectAsID** instance. The object instance can be retrieved with @GlobalScope.instance_from_id.

ENetConnection

Inherits: RefCounted < Object

A wrapper class for an ENetHost [http://enet.bespin.org/group__host.html].

Description

ENet's purpose is to provide a relatively thin, simple and robust network communication layer on top of UDP (User Datagram Protocol).

Tutorials

- API documentation on the ENet website [http://enet.bespin.org/usergroup0.html]

Methods

void	bandwidth_limit(in_bandwidth: int = 0, out_bandwidth: int = 0)
void	broadcast(channel: int, packet: PackedByteArray, flags: int)
void	channel_limit(limit: int)

void	compress(mode: CompressionMode)
ENetPacketPeer	connect_to_host(address: String, port: int, channels: int = 0, data: int = 0)
Error	create_host(max_peers: int = 32, max_channels: int = 0, in_bandwidth: int = 0, out_bandwidth: int = 0)
Error	create_host_bound(bind_address: String, bind_port: int, max_peers: int = 32, max_channels: int = 0, in_bandwidth: int = 0, out_bandwidth: int = 0)
void	destroy()
Error	dtls_client_setup(hostname: String, client_options: TLSOptions = null)
Error	dtls_server_setup(server_options: TLSOptions)
void	flush()
int	get_local_port() const
int	get_max_channels() const

Array[ENetPacketPeer]	get_peers()
float	pop_statistic(statistic: HostStatistic)
void	refuse_new_connections(refuse: bool)
Array	service(timeout: int = 0)
void	socket_send(destination_address: String, destination_port: int, packet: PackedByteArray)

Enumerations

enum **CompressionMode**: 🔗

CompressionMode **COMPRESS_NONE** = `0`

No compression. This uses the most bandwidth, but has the upside of requiring the fewest CPU resources. This option may also be used to make network debugging using tools like Wireshark easier.

CompressionMode **COMPRESS_RANGE_CODER** = `1`

ENet's built-in range encoding. Works well on small packets, but is not the most efficient algorithm on packets larger than 4 KB.

CompressionMode **COMPRESS_FASTLZ** = `2`

FastLZ [https://fastlz.org/] compression. This option uses less CPU resources compared to COMPRESS_ZLIB, at the expense of using more bandwidth.

CompressionMode **COMPRESS_ZLIB** = `3`

Zlib [https://www.zlib.net/] compression. This option uses less bandwidth compared to COMPRESS_FASTLZ, at the expense of using more CPU resources.

CompressionMode **COMPRESS_ZSTD** = `4`

Zstandard [https://facebook.github.io/zstd/] compression. Note that this algorithm is not very efficient on packets smaller than 4 KB. Therefore, it's recommended to use other compression algorithms in most cases.

---

enum **EventType**: 🔗

EventType **EVENT_ERROR** = `-1`

An error occurred during service. You will likely need to destroy the host and recreate it.

EventType **EVENT_NONE** = `0`

No event occurred within the specified time limit.

EventType **EVENT_CONNECT** = `1`

A connection request initiated by enet_host_connect has completed. The array will contain the peer which successfully connected.

EventType **EVENT_DISCONNECT** = `2`

A peer has disconnected. This event is generated on a successful completion of a disconnect initiated by ENetPacketPeer.peer_disconnect, if a peer has timed out, or if a connection request initialized by connect_to_host has timed out. The array will contain the peer which disconnected. The data field contains user supplied data describing the disconnection, or 0, if none is available.

EventType **EVENT_RECEIVE** = `3`

A packet has been received from a peer. The array will contain the peer which sent the packet and the channel number upon which the packet was received. The received packet will be queued to the associated ENetPacketPeer.

enum **HostStatistic**: 🔗

HostStatistic **HOST_TOTAL_SENT_DATA** = `0`

Total data sent.

HostStatistic **HOST_TOTAL_SENT_PACKETS** = `1`

Total UDP packets sent.

HostStatistic **HOST_TOTAL_RECEIVED_DATA** = `2`

Total data received.

HostStatistic **HOST_TOTAL_RECEIVED_PACKETS** = `3`

Total UDP packets received.

Method Descriptions

void **bandwidth_limit**(in_bandwidth: int = 0, out_bandwidth: int = 0)

Adjusts the bandwidth limits of a host.

---

void **broadcast**(channel: int, packet: PackedByteArray, flags: int)

Queues a `packet` to be sent to all peers associated with the host over the specified `channel`. See ENetPacketPeer `FLAG_*` constants for available packet flags.

---

void **channel_limit**(limit: int)

Limits the maximum allowed channels of future incoming connections.

---

void **compress**(mode: CompressionMode)

Sets the compression method used for network packets. These have different tradeoffs of compression speed versus bandwidth, you may need to test which one works best for your use case if you use compression at all.

Note: Most games' network design involve sending many small packets frequently (smaller than 4 KB each). If in doubt, it is recommended to keep the default compression algorithm as it works best on these small packets.

Note: The compression mode must be set to the same value on both the server and all its clients. Clients will fail to connect if the compression mode set on the client differs from the one set on the server.

ENetPacketPeer **connect_to_host**(address: String, port: int, channels: int = 0, data: int = 0)

Initiates a connection to a foreign `address` using the specified `port` and allocating the requested `channels`. Optional `data` can be passed during connection in the form of a 32 bit integer.

Note: You must call either create_host or create_host_bound on both ends before calling this method.

Error **create_host**(max_peers: int = 32, max_channels: int = 0, in_bandwidth: int = 0, out_bandwidth: int = 0)

Creates an ENetHost that allows up to `max_peers` connected peers, each allocating up to `max_channels` channels, optionally limiting bandwidth to `in_bandwidth` and `out_bandwidth` (if greater than zero).

This method binds a random available dynamic UDP port on the host machine at the *unspecified* address. Use create_host_bound to specify the address and port.

Note: It is necessary to create a host in both client and server in order to establish a connection.

Error **create_host_bound**(bind_address: String, bind_port: int, max_peers: int = 32, max_channels: int = 0, in_bandwidth: int = 0, out_bandwidth: int = 0)

Creates an ENetHost bound to the given `bind_address` and `bind_port` that allows up to `max_peers` connected peers, each allocating up to `max_channels` channels, optionally limiting bandwidth to `in_bandwidth` and `out_bandwidth` (if greater than zero).

Note: It is necessary to create a host in both client and server in order to establish a connection.

---

void **destroy**()

Destroys the host and all resources associated with it.

Error **dtls_client_setup**(hostname: String, client_options: TLSOptions = null)

Configure this ENetHost to use the custom Godot extension allowing DTLS encryption for ENet clients. Call this before connect_to_host to have ENet connect using DTLS validating the server certificate against `hostname`. You can pass the optional `client_options` parameter to customize the trusted certification authorities, or disable the common name verification. See TLSOptions.client and TLSOptions.client_unsafe.

Error **dtls_server_setup**(server_options: TLSOptions)

Configure this ENetHost to use the custom Godot extension allowing DTLS encryption for ENet servers. Call this right after create_host_bound to have ENet expect peers to connect using DTLS. See TLSOptions.server.

---

void **flush**()

Sends any queued packets on the host specified to its designated peers.

---

int **get_local_port**() const

Returns the local port to which this peer is bound.

---

int **get_max_channels**() const 🔗

Returns the maximum number of channels allowed for connected peers.

Array[ENetPacketPeer] **get_peers**() 🔗

Returns the list of peers associated with this host.

Note: This list might include some peers that are not fully connected or are still being disconnected.

---

float **pop_statistic**(statistic: HostStatistic) 🔗

Returns and resets host statistics. See HostStatistic for more info.

---

void **refuse_new_connections**(refuse: bool) 🔗

Configures the DTLS server to automatically drop new connections.

Note: This method is only relevant after calling dtls_server_setup.

Array **service**(timeout: int = 0) 🔗

Waits for events on this connection and shuttles packets between the host and its peers, with the given `timeout` (in milliseconds). The returned Array will have 4 elements. An EventType, the ENetPacketPeer which generated the event, the event associated data (if any), the event associated channel (if any). If the generated event is EVENT_RECEIVE, the received packet will be queued to the associated ENetPacketPeer.

Call this function regularly to handle connections, disconnections, and to receive new packets.

Note: This method must be called on both ends involved in the event (sending and receiving hosts).

---

void **socket_send**(destination_address: String, destination_port: int, packet: PackedByteArray) 🔗

Sends a `packet` toward a destination from the address and port currently bound by this ENetConnection instance.

This is useful as it serves to establish entries in NAT routing tables on all devices between this bound instance and the public facing internet, allowing a prospective client's connection packets to be routed backward through the NAT device(s) between the public internet and this host.

This requires forward knowledge of a prospective client's address and communication port as seen by the public internet - after any NAT devices have handled their connection request. This information can be obtained by a STUN [https://en.wikipedia.org/wiki/STUN] service, and must be handed off to your host by an entity that is not the prospective client. This will never work for a client behind a Symmetric NAT due to the nature of the Symmetric NAT routing algorithm, as their IP and Port cannot be known beforehand.

ENetMultiplayerPeer

Inherits: MultiplayerPeer **<** PacketPeer **<** RefCounted **<** Object

A MultiplayerPeer implementation using the ENet [http://enet.bespin.org/index.html] library.

Description

A MultiplayerPeer implementation that should be passed to MultiplayerAPI.multiplayer_peer after being initialized as either a client, server, or mesh. Events can then be handled by connecting to MultiplayerAPI signals. See ENetConnection for more information on the ENet library wrapper.

Note: ENet only uses UDP, not TCP. When forwarding the server port to make your server accessible on the public Internet, you only need to forward the server port in UDP. You can use the UPNP class to try to forward the server port automatically when starting the server.

Tutorials

- High-level multiplayer
- API documentation on the ENet website [http://enet.bespin.org/usergroup0.html]

Properties

ENetConnection	host

Methods

Error	add_mesh_peer(peer_id: int, host: ENetConnection)
Error	create_client(address: String, port: int, channel_count: int = 0, in_bandwidth: int = 0, out_bandwidth: int = 0, local_port: int = 0)
Error	create_mesh(unique_id: int)
Error	create_server(port: int, max_clients: int = 32, max_channels: int = 0, in_bandwidth: int = 0, out_bandwidth: int = 0)
ENetPacketPeer	get_peer(id: int) const
void	set_bind_ip(ip: String)

Property Descriptions

ENetConnection **host** 🔗

- ENetConnection **get_host**()

The underlying ENetConnection created after create_client and create_server.

Method Descriptions

Error **add_mesh_peer**(peer_id: int, host: ENetConnection) 🔗

Add a new remote peer with the given `peer_id` connected to the given `host`.

Note: The `host` must have exactly one peer in the ENetPacketPeer.STATE_CONNECTED state.

Error **create_client**(address: String, port: int, channel_count: int = 0, in_bandwidth: int = 0, out_bandwidth: int = 0, local_port: int = 0) 🔗

Create client that connects to a server at `address` using specified `port`. The given address needs to be either a fully qualified domain name (e.g. `"www.example.com"`) or an IP address in IPv4 or IPv6 format (e.g. `"192.168.1.1"`). The `port` is the port the server is listening on. The `channel_count` parameter can be used to specify the number of ENet channels allocated for the connection. The `in_bandwidth` and `out_bandwidth` parameters can be used to limit the incoming and outgoing bandwidth to the given number of bytes per second. The default of 0 means unlimited bandwidth. Note that ENet will strategically drop packets on specific sides of a connection between peers to ensure the peer's bandwidth is not overwhelmed. The bandwidth parameters also determine the window size of a connection which limits the amount of reliable packets that may be in transit at any given time. Returns @GlobalScope.OK if a client was created, @GlobalScope.ERR_ALREADY_IN_USE if this ENetMultiplayerPeer instance already has an open connection (in which case you need to call MultiplayerPeer.close first) or @GlobalScope.ERR_CANT_CREATE if the client could not be created. If `local_port` is specified, the client will also listen to the given port; this is useful for some NAT traversal techniques.

Error **create_mesh**(unique_id: int)

Initialize this MultiplayerPeer in mesh mode. The provided `unique_id` will be used as the local peer network unique ID once assigned as the MultiplayerAPI.multiplayer_peer. In the mesh configuration you will need to set up each new peer manually using ENetConnection before calling add_mesh_peer. While this technique is more advanced, it allows for better control over the connection process (e.g. when dealing with NAT punch-through) and for better distribution of the network load (which would otherwise be more taxing on the server).

Error **create_server**(port: int, max_clients: int = 32, max_channels: int = 0, in_bandwidth: int = 0, out_bandwidth: int = 0)

Create server that listens to connections via `port`. The port needs to be an available, unused port between 0 and 65535. Note that ports below 1024 are privileged and may require elevated permissions depending on the platform. To change the interface the server listens on, use set_bind_ip. The default IP is the wildcard `"*"`, which listens on all available interfaces. `max_clients` is the maximum number of clients that are allowed at once, any number up to 4095 may be used, although the achievable number of simultaneous clients may be far lower and depends on the application. For additional details on the bandwidth parameters, see create_client. Returns @GlobalScope.OK if a server was created, @GlobalScope.ERR_ALREADY_IN_USE if this ENetMultiplayerPeer instance already has an open connection (in which case you need to call MultiplayerPeer.close first) or @GlobalScope.ERR_CANT_CREATE if the server could not be created.

ENetPacketPeer **get_peer**(id: int) const 🔗

Returns the ENetPacketPeer associated to the given `id`.

---

void **set_bind_ip**(ip: String) 🔗

The IP used when creating a server. This is set to the wildcard `"*"` by default, which binds to all available interfaces. The given IP needs to be in IPv4 or IPv6 address format, for example: `"192.168.1.1"`.

ENetPacketPeer

Inherits: PacketPeer **<** RefCounted **<** Object

A wrapper class for an ENetPeer [http://enet.bespin.org/group__peer.html].

Description

A PacketPeer implementation representing a peer of an ENetConnection.

This class cannot be instantiated directly but can be retrieved during ENetConnection.service or via ENetConnection.get_peers.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Tutorials

- API documentation on the ENet website [http://enet.bespin.org/usergroup0.html]

Methods

int	get_channels() const
String	get_remote_address() const

int	get_remote_port() const
PeerState	get_state() const
float	get_statistic(statistic: PeerStatistic)
bool	is_active() const
void	peer_disconnect(data: int = 0)
void	peer_disconnect_later(data: int = 0)
void	peer_disconnect_now(data: int = 0)
void	ping()
void	ping_interval(ping_interval: int)
void	reset()
Error	send(channel: int, packet: PackedByteArray, flags: int)
void	set_timeout(timeout: int, timeout_min: int, timeout_max: int)

void	throttle_configure(interval: int, acceleration: int, deceleration: int)

Enumerations

enum **PeerState**: 🔗

PeerState **STATE_DISCONNECTED** = `0`

The peer is disconnected.

PeerState **STATE_CONNECTING** = `1`

The peer is currently attempting to connect.

PeerState **STATE_ACKNOWLEDGING_CONNECT** = `2`

The peer has acknowledged the connection request.

PeerState **STATE_CONNECTION_PENDING** = `3`

The peer is currently connecting.

PeerState **STATE_CONNECTION_SUCCEEDED** = `4`

The peer has successfully connected, but is not ready to communicate with yet (STATE_CONNECTED).

PeerState **STATE_CONNECTED** = `5`

The peer is currently connected and ready to communicate with.

PeerState **STATE_DISCONNECT_LATER** = `6`

The peer is slated to disconnect after it has no more outgoing packets to send.

PeerState **STATE_DISCONNECTING** = `7`

The peer is currently disconnecting.

PeerState **STATE_ACKNOWLEDGING_DISCONNECT** = `8`

The peer has acknowledged the disconnection request.

PeerState **STATE_ZOMBIE** = `9`

The peer has lost connection, but is not considered truly disconnected (as the peer didn't acknowledge the disconnection request).

---

enum **PeerStatistic**: 🔗

PeerStatistic **PEER_PACKET_LOSS** = `0`

Mean packet loss of reliable packets as a ratio with respect to the PACKET_LOSS_SCALE.

PeerStatistic **PEER_PACKET_LOSS_VARIANCE** = `1`

Packet loss variance.

PeerStatistic **PEER_PACKET_LOSS_EPOCH** = `2`

The time at which packet loss statistics were last updated (in milliseconds since the connection started). The interval for packet loss statistics updates is 10 seconds, and at least one packet must have been sent since the last statistics update.

PeerStatistic **PEER_ROUND_TRIP_TIME** = `3`

Mean packet round trip time for reliable packets.

PeerStatistic **PEER_ROUND_TRIP_TIME_VARIANCE** = `4`

Variance of the mean round trip time.

PeerStatistic **PEER_LAST_ROUND_TRIP_TIME** = `5`

Last recorded round trip time for a reliable packet.

PeerStatistic **PEER_LAST_ROUND_TRIP_TIME_VARIANCE** = `6`

Variance of the last trip time recorded.

PeerStatistic **PEER_PACKET_THROTTLE** = `7`

The peer's current throttle status.

PeerStatistic **PEER_PACKET_THROTTLE_LIMIT** = `8`

The maximum number of unreliable packets that should not be dropped. This value is always greater than or equal to `1`. The initial value is equal to PACKET_THROTTLE_SCALE.

PeerStatistic **PEER_PACKET_THROTTLE_COUNTER** = `9`

Internal value used to increment the packet throttle counter. The value is hardcoded to `7` and cannot be changed. You probably want to look at PEER_PACKET_THROTTLE_ACCELERATION instead.

PeerStatistic **PEER_PACKET_THROTTLE_EPOCH** = `10`

The time at which throttle statistics were last updated (in milliseconds since the connection started). The interval for throttle statistics updates is PEER_PACKET_THROTTLE_INTERVAL.

PeerStatistic **PEER_PACKET_THROTTLE_ACCELERATION** = `11`

The throttle's acceleration factor. Higher values will make ENet adapt to fluctuating network conditions faster, causing unrelaible packets to be sent *more* often. The default value is `2`.

PeerStatistic **PEER_PACKET_THROTTLE_DECELERATION** = `12`

The throttle's deceleration factor. Higher values will make ENet adapt to fluctuating network conditions faster, causing unrelaible packets to be sent *less* often. The default value is `2`.

PeerStatistic **PEER_PACKET_THROTTLE_INTERVAL** = `13`

The interval over which the lowest mean round trip time should be measured for use by the throttle mechanism (in milliseconds). The default value is `5000`.

Constants

PACKET_LOSS_SCALE = `65536`

The reference scale for packet loss. See get_statistic and PEER_PACKET_LOSS.

PACKET_THROTTLE_SCALE = `32`

The reference value for throttle configuration. The default value is `32`. See throttle_configure.

FLAG_RELIABLE = `1`

Mark the packet to be sent as reliable.

FLAG_UNSEQUENCED = `2`

Mark the packet to be sent unsequenced (unreliable).

FLAG_UNRELIABLE_FRAGMENT = `8` 🔗

Mark the packet to be sent unreliable even if the packet is too big and needs fragmentation (increasing the chance of it being dropped).

Method Descriptions

int **get_channels**() const 🔗

Returns the number of channels allocated for communication with peer.

String **get_remote_address**() const 🔗

Returns the IP address of this peer.

---

int **get_remote_port**() const 🔗

Returns the remote port of this peer.

PeerState **get_state**() const 🔗

Returns the current peer state. See PeerState.

---

float **get_statistic**(statistic: PeerStatistic) 🔗

Returns the requested `statistic` for this peer. See PeerStatistic.

---

bool **is_active**() const 🔗

Returns `true` if the peer is currently active (i.e. the associated ENetConnection is still valid).

void **peer_disconnect**(data: int = 0)

Request a disconnection from a peer. An ENetConnection.EVENT_DISCONNECT will be generated during ENetConnection.service once the disconnection is complete.

void **peer_disconnect_later**(data: int = 0)

Request a disconnection from a peer, but only after all queued outgoing packets are sent. An ENetConnection.EVENT_DISCONNECT will be generated during ENetConnection.service once the disconnection is complete.

void **peer_disconnect_now**(data: int = 0)

Force an immediate disconnection from a peer. No ENetConnection.EVENT_DISCONNECT will be generated. The foreign peer is not guaranteed to receive the disconnect notification, and is reset immediately upon return from this function.

void **ping**()

Sends a ping request to a peer. ENet automatically pings all connected peers at regular intervals, however, this function may be called to ensure more frequent ping requests.

void **ping_interval**(ping_interval: int)

Sets the `ping_interval` in milliseconds at which pings will be sent to a peer. Pings are used both to monitor the liveness of the connection and also to dynamically adjust the throttle during periods of low traffic so that the throttle has reasonable responsiveness during traffic spikes. The default ping interval is `500` milliseconds.

---

void **reset**()

Forcefully disconnects a peer. The foreign host represented by the peer is not notified of the disconnection and will timeout on its connection to the local host.

Error **send**(channel: int, packet: PackedByteArray, flags: int)

Queues a `packet` to be sent over the specified `channel`. See `FLAG_*` constants for available packet flags.

---

void **set_timeout**(timeout: int, timeout_min: int, timeout_max: int)

Sets the timeout parameters for a peer. The timeout parameters control how and when a peer will timeout from a failure to acknowledge reliable traffic. Timeout values are expressed in milliseconds.

The `timeout` is a factor that, multiplied by a value based on the average round trip time, will determine the timeout limit for a reliable packet. When that limit is reached, the timeout will be doubled, and the peer will be disconnected if that limit has reached `timeout_min`. The `timeout_max` parameter, on the other hand, defines a fixed timeout for which any packet must be acknowledged or the peer will be dropped.

---

void **throttle_configure**(interval: int, acceleration: int, deceleration: int) 🔗

Configures throttle parameter for a peer.

Unreliable packets are dropped by ENet in response to the varying conditions of the Internet connection to the peer. The throttle represents a probability that an unreliable packet should not be dropped and thus sent by ENet to the peer. By measuring fluctuations in round trip times of reliable packets over the specified `interval`, ENet will either increase the probability by the amount specified in the `acceleration` parameter, or decrease it by the amount specified in the `deceleration` parameter (both are ratios to PACKET_THROTTLE_SCALE).

When the throttle has a value of PACKET_THROTTLE_SCALE, no unreliable packets are dropped by ENet, and so 100% of all unreliable packets will be sent.

When the throttle has a value of `0`, all unreliable packets are dropped by ENet, and so 0% of all unreliable packets will be sent.

Intermediate values for the throttle represent intermediate probabilities between 0% and 100% of unreliable packets being sent. The bandwidth limits of the local and foreign hosts are taken into account to determine a sensible limit for the throttle probability above which it should not raise even in the best of conditions.

Engine

Inherits: Object

Provides access to engine properties.

Description

The **Engine** singleton allows you to query and modify the project's run-time parameters, such as frames per second, time scale, and others. It also stores information about the current build of Godot, such as the current version.

Properties

int	max_fps	`0`
int	max_physics_steps_per_frame	`8`
float	physics_jitter_fix	`0.5`
int	physics_ticks_per_second	`60`
bool	print_error_messages	`true`

float	time_scale	`1.0`

Methods

String	get_architecture_name() const
Dictionary	get_author_info() const
Array[Dictionary]	get_copyright_info() const
Dictionary	get_donor_info() const
int	get_frames_drawn()
float	get_frames_per_second() const
Dictionary	get_license_info() const
String	get_license_text() const
MainLoop	get_main_loop() const
int	get_physics_frames() const

float	get_physics_interpolation_fraction() const
int	get_process_frames() const
ScriptLanguage	get_script_language(index: int) const
int	get_script_language_count()
Object	get_singleton(name: StringName) const
PackedStringArray	get_singleton_list() const
Dictionary	get_version_info() const
String	get_write_movie_path() const
bool	has_singleton(name: StringName) const
bool	is_editor_hint() const
bool	is_in_physics_frame() const
Error	register_script_language(language: ScriptLanguage)

void	register_singleton(name: StringName, instance: Object)
Error	unregister_script_language(language: ScriptLanguage)
void	unregister_singleton(name: StringName)

Property Descriptions

int **max_fps** = `0` 🔗

- void **set_max_fps**(value: int)
- int **get_max_fps**()

The maximum number of frames that can be rendered every second (FPS). A value of `0` means the framerate is uncapped.

Limiting the FPS can be useful to reduce the host machine's power consumption, which reduces heat, noise emissions, and improves battery life.

If ProjectSettings.display/window/vsync/vsync_mode is **Enabled** or **Adaptive**, the setting takes precedence and the max FPS number cannot exceed the monitor's refresh rate.

If ProjectSettings.display/window/vsync/vsync_mode is **Enabled**, on monitors with variable refresh rate enabled (G-Sync/FreeSync), using an FPS limit a few frames lower than the monitor's refresh rate will reduce input lag while avoiding tearing [https://blurbusters.com/howto-low-lag-vsync-on/].

See also physics_ticks_per_second and ProjectSettings.application/run/max_fps.

Note: The actual number of frames per second may still be below this value if the CPU or GPU cannot keep up with the project's logic and rendering.

Note: If ProjectSettings.display/window/vsync/vsync_mode is **Disabled**, limiting the FPS to a high value that can be consistently reached on the system can reduce input lag compared to an uncapped framerate. Since this works by ensuring the GPU load is lower than 100%, this latency reduction is only effective in GPU-bottlenecked scenarios, not CPU-bottlenecked scenarios.

---

int **max_physics_steps_per_frame** = `8` 🔗

- void **set_max_physics_steps_per_frame**(value: int)
- int **get_max_physics_steps_per_frame**()

The maximum number of physics steps that can be simulated each rendered frame.

Note: The default value is tuned to prevent expensive physics simulations from triggering even more expensive simulations indefinitely. However, the game will appear to slow down if the rendering FPS is less than `1 / max_physics_steps_per_frame` of physics_ticks_per_second. This occurs even if `delta` is consistently used in physics calculations. To avoid this, increase max_physics_steps_per_frame if you have increased physics_ticks_per_second significantly above its default value.

---

float **physics_jitter_fix** = `0.5` 🔗

- void **set_physics_jitter_fix**(value: float)
- float **get_physics_jitter_fix**()

How much physics ticks are synchronized with real time. If `0` or less, the ticks are fully synchronized. Higher values cause the in-game clock to deviate more from the real clock, but they smooth out framerate jitters.

Note: The default value of `0.5` should be good enough for most cases; values above `2` could cause the game to react to dropped frames with a noticeable delay and are not recommended.

Note: When using a custom physics interpolation solution, or within a network game, it's recommended to disable the physics jitter fix by setting this property to `0`.

---

int **physics_ticks_per_second** = `60` 🔗

- void **set_physics_ticks_per_second**(value: int)
- int **get_physics_ticks_per_second**()

The number of fixed iterations per second. This controls how often physics simulation and Node._physics_process methods are run. This value should generally always be set to `60` or above, as Godot doesn't interpolate the physics step. As a result, values lower than `60` will look stuttery. This value can be increased to make input more reactive or work around collision tunneling issues, but keep in mind doing so will increase CPU usage. See also max_fps and ProjectSettings.physics/common/physics_ticks_per_second.

Note: Only max_physics_steps_per_frame physics ticks may be simulated per rendered frame at most. If more physics ticks have to be simulated per rendered frame to keep up with rendering, the project will appear to slow down (even if `delta` is used consistently in physics calculations). Therefore, it is recommended to also increase max_physics_steps_per_frame if increasing physics_ticks_per_second significantly above its default value.

---

bool **print_error_messages** = `true` 🔗

- void **set_print_error_messages**(value: bool)
- bool **is_printing_error_messages**()

If `false`, stops printing error and warning messages to the console and editor Output log. This can be used to hide error and warning messages during unit test suite runs. This property is equivalent to the ProjectSettings.application/run/disable_stderr project setting.

Note: This property does not impact the editor's Errors tab when running a project from the editor.

Warning: If set to `false` anywhere in the project, important error messages may be hidden even if they are emitted from other scripts. In a `@tool` script, this will also impact the editor itself. Do *not* report bugs before ensuring error messages are enabled (as they are by default).

---

float **time_scale** = `1.0` 🔗

- void **set_time_scale**(value: float)
- float **get_time_scale**()

The speed multiplier at which the in-game clock updates, compared to real time. For example, if set to `2.0` the game runs twice as fast, and if set to `0.5` the game runs half as fast.

This value affects Timer, SceneTreeTimer, and all other simulations that make use of `delta` time (such as Node._process and Node._physics_process).

Note: It's recommended to keep this property above `0.0`, as the game may behave unexpectedly otherwise.

Note: This does not affect audio playback speed. Use AudioServer.playback_speed_scale to adjust audio playback speed independently of time_scale.

Note: This does not automatically adjust physics_ticks_per_second. With values above `1.0` physics simulation may become less precise, as each physics tick will stretch over a larger period of engine time. If you're modifying time_scale to speed up simulation by a large factor, consider also increasing physics_ticks_per_second to make the simulation more reliable.

Method Descriptions

String **get_architecture_name**() const

Returns the name of the CPU architecture the Godot binary was built for. Possible return values include `"x86_64"`, `"x86_32"`, `"arm64"`, `"arm32"`, `"rv64"`, `"riscv"`, `"ppc64"`, `"ppc"`, `"wasm64"`, and `"wasm32"`.

To detect whether the current build is 64-bit, you can use the fact that all 64-bit architecture names contain `64` in their name:

GDScript

```
if "64" in Engine.get_architecture_name():
    print("Running a 64-bit build of Godot.")
else:
    print("Running a 32-bit build of Godot.")
```

C#

```
if (Engine.GetArchitectureName().Contains("64"))
    GD.Print("Running a 64-bit build of Godot.");
else
    GD.Print("Running a 32-bit build of Godot.");
```

Note: This method does *not* return the name of the system's CPU architecture (like OS.get_processor_name). For example, when running an `x86_32` Godot binary on an `x86_64` system, the returned value will still be `"x86_32"`.

Dictionary **get_author_info**() const 🔗

Returns the engine author information as a Dictionary, where each entry is an Array of strings with the names of notable contributors to the Godot Engine: `lead_developers`, `founders`, `project_managers`, and `developers`.

Array[Dictionary] **get_copyright_info**() const 🔗

Returns an Array of dictionaries with copyright information for every component of Godot's source code.

Every Dictionary contains a `name` identifier, and a `parts` array of dictionaries. It describes the component in detail with the following entries:

- `files` - Array of file paths from the source code affected by this component;
- `copyright` - Array of owners of this component;
- `license` - The license applied to this component (such as "Expat [https://en.wikipedia.org/wiki/MIT_License#Ambiguity_and_variants]" or "CC-BY-4.0 [https://creativecommons.org/licenses/by/4.0/]").

Dictionary **get_donor_info**() const 🔗

Returns a Dictionary of categorized donor names. Each entry is an Array of strings:

{`platinum_sponsors`, `gold_sponsors`, `silver_sponsors`, `bronze_sponsors`, `mini_sponsors`, `gold_donors`, `silver_donors`, `bronze_donors`}

int **get_frames_drawn**()

Returns the total number of frames drawn since the engine started.

Note: On headless platforms, or if rendering is disabled with `--disable-render-loop` via command line, this method always returns `0`. See also get_process_frames.

float **get_frames_per_second**() const

Returns the average frames rendered every second (FPS), also known as the framerate.

Dictionary **get_license_info**() const

Returns a Dictionary of licenses used by Godot and included third party components. Each entry is a license name (such as "Expat [https://en.wikipedia.org/wiki/MIT_License#Ambiguity_and_variants]") and its associated text.

String **get_license_text**() const

Returns the full Godot license text.

MainLoop **get_main_loop**() const

Returns the instance of the MainLoop. This is usually the main SceneTree and is the same as Node.get_tree.

Note: The type instantiated as the main loop can changed with ProjectSettings.application/run/main_loop_type.

---

int **get_physics_frames**() const

Returns the total number of frames passed since the engine started. This number is increased every **physics frame**. See also get_process_frames.

This method can be used to run expensive logic less often without relying on a Timer:

GDScript

```
func _physics_process(_delta):
    if Engine.get_physics_frames() % 2 == 0:
        pass # Run expensive logic only once every 2 physics frames here.
```

C#

```
public override void _PhysicsProcess(double delta)
{
    base._PhysicsProcess(delta);

    if (Engine.GetPhysicsFrames() % 2 == 0)
    {
        // Run expensive logic only once every 2 physics frames here.
    }
}
```

---

float **get_physics_interpolation_fraction**() const

Returns the fraction through the current physics tick we are at the time of rendering the frame. This can be used to implement fixed timestep interpolation.

---

int **get_process_frames**() const 🔗

Returns the total number of frames passed since the engine started. This number is increased every **process frame**, regardless of whether the render loop is enabled. See also get_frames_drawn and get_physics_frames.

This method can be used to run expensive logic less often without relying on a Timer:

GDScript

```gdscript
func _process(_delta):
    if Engine.get_process_frames() % 5 == 0:
        pass # Run expensive logic only once every 5 process (render) frames here.
```

C#

```csharp
public override void _Process(double delta)
{
    base._Process(delta);

    if (Engine.GetProcessFrames() % 5 == 0)
    {
        // Run expensive logic only once every 5 process (render) frames here.
    }
}
```

ScriptLanguage **get_script_language**(index: int) const 🔗

Returns an instance of a ScriptLanguage with the given `index`.

---

int **get_script_language_count**() 🔗

Returns the number of available script languages. Use with get_script_language.

Object **get_singleton**(name: StringName) const 🔗

Returns the global singleton with the given `name`, or `null` if it does not exist. Often used for plugins. See also has_singleton and get_singleton_list.

Note: Global singletons are not the same as autoloaded nodes, which are configurable in the project settings.

PackedStringArray **get_singleton_list**() const 🔗

Returns a list of names of all available global singletons. See also get_singleton.

Dictionary **get_version_info**() const 🔗

Returns the current engine version information as a Dictionary containing the following entries:

- `major` - Major version number as an int;
- `minor` - Minor version number as an int;
- `patch` - Patch version number as an int;
- `hex` - Full version encoded as a hexadecimal int with one byte (2 hex digits) per number (see example below);
- `status` - Status (such as "beta", "rc1", "rc2", "stable", etc.) as a String;
- `build` - Build name (e.g. "custom_build") as a String;

- `hash` - Full Git commit hash as a String;
- `timestamp` - Holds the Git commit date UNIX timestamp in seconds as an int, or `0` if unavailable;
- `string` - `major`, `minor`, `patch`, `status`, and `build` in a single String.

The `hex` value is encoded as follows, from left to right: one byte for the major, one byte for the minor, one byte for the patch version. For example, "3.1.12" would be `0x03010C`.

Note: The `hex` value is still an int internally, and printing it will give you its decimal representation, which is not particularly meaningful. Use hexadecimal literals for quick version comparisons from code:

GDScript

```
if Engine.get_version_info().hex >= 0x040100:
    pass # Do things specific to version 4.1 or later.
else:
    pass # Do things specific to versions before 4.1.
```

C#

```
if ((int)Engine.GetVersionInfo()["hex"] >= 0x040100)
{
    // Do things specific to version 4.1 or later.
}
else
{
    // Do things specific to versions before 4.1.
}
```

String **get_write_movie_path**() const 🔗

Returns the path to the MovieWriter's output file, or an empty string if the engine wasn't started in Movie Maker mode. The default path can be changed in ProjectSettings.editor/movie_writer/movie_file.

---

bool **has_singleton**(name: StringName) const

Returns `true` if a singleton with the given `name` exists in the global scope. See also get_singleton.

GDScript

```
print(Engine.has_singleton("OS"))          # Prints true
print(Engine.has_singleton("Engine"))      # Prints true
print(Engine.has_singleton("AudioServer")) # Prints true
print(Engine.has_singleton("Unknown"))     # Prints false
```

C#

```
GD.Print(Engine.HasSingleton("OS"));          // Prints true
GD.Print(Engine.HasSingleton("Engine"));      // Prints true
GD.Print(Engine.HasSingleton("AudioServer")); // Prints true
GD.Print(Engine.HasSingleton("Unknown"));     // Prints false
```

Note: Global singletons are not the same as autoloaded nodes, which are configurable in the project settings.

---

bool **is_editor_hint**() const

Returns `true` if the script is currently running inside the editor, otherwise returns `false`. This is useful for `@tool` scripts to conditionally draw editor helpers, or prevent accidentally running "game" code that would affect the scene state while in the editor:

GDScript

```
if Engine.is_editor_hint():
    draw_gizmos()
else:
    simulate_physics()
```

C#

```
if (Engine.IsEditorHint())
    DrawGizmos();
else
    SimulatePhysics();
```

See Running code in the editor in the documentation for more information.

Note: To detect whether the script is running on an editor *build* (such as when pressing `F5`), use OS.has_feature with the `"editor"` argument instead. `OS.has_feature("editor")` evaluate to `true` both when the script is running in the editor and when running the project from the editor, but returns `false` when run from an exported project.

---

bool **is_in_physics_frame**() const 🔗

Returns `true` if the engine is inside the fixed physics process step of the main loop.

```
func _enter_tree():
    # Depending on when the node is added to the tree,
    # prints either "true" or "false".
    print(Engine.is_in_physics_frame())

func _process(delta):
    print(Engine.is_in_physics_frame()) # Prints false

func _physics_process(delta):
    print(Engine.is_in_physics_frame()) # Prints true
```

Error **register_script_language**(language: ScriptLanguage) 🔗

Registers a ScriptLanguage instance to be available with `ScriptServer`.

Returns:

- @GlobalScope.OK on success;
- @GlobalScope.ERR_UNAVAILABLE if `ScriptServer` has reached the limit and cannot register any new language;
- @GlobalScope.ERR_ALREADY_EXISTS if `ScriptServer` already contains a language with similar extension/name/type.

---

void **register_singleton**(name: StringName, instance: Object)

Registers the given Object `instance` as a singleton, available globally under `name`. Useful for plugins.

Error **unregister_script_language**(language: ScriptLanguage)

Unregisters the ScriptLanguage instance from `ScriptServer`.

Returns:

- @GlobalScope.OK on success;
- @GlobalScope.ERR_DOES_NOT_EXIST if the language is not registered in `ScriptServer`.

---

void **unregister_singleton**(name: StringName)

Removes the singleton registered under `name`. The singleton object is *not* freed. Only works with user-defined singletons registered with register_singleton.

EngineDebugger

Inherits: Object

Exposes the internal debugger.

Description

EngineDebugger handles the communication between the editor and the running game. It is active in the running game. Messages can be sent/received through it. It also manages the profilers.

Methods

void	clear_breakpoints()
void	debug(can_continue: bool = true, is_error_breakpoint: bool = false)
int	get_depth() const
int	get_lines_left() const
bool	has_capture(name: StringName)
bool	has_profiler(name: StringName)

void	insert_breakpoint(line: int, source: StringName)
bool	is_active()
bool	is_breakpoint(line: int, source: StringName) const
bool	is_profiling(name: StringName)
bool	is_skipping_breakpoints() const
void	line_poll()
void	profiler_add_frame_data(name: StringName, data: Array)
void	profiler_enable(name: StringName, enable: bool, arguments: Array = [])
void	register_message_capture(name: StringName, callable: Callable)
void	register_profiler(name: StringName, profiler: EngineProfiler)
void	remove_breakpoint(line: int, source: StringName)
void	script_debug(language: ScriptLanguage, can_continue: bool = true, is_error_breakpoint: bool = false)

void	send_message(message: String, data: Array)
void	set_depth(depth: int)
void	set_lines_left(lines: int)
void	unregister_message_capture(name: StringName)
void	unregister_profiler(name: StringName)

Method Descriptions

void **clear_breakpoints**() 🔗

Clears all breakpoints.

---

void **debug**(can_continue: bool = true, is_error_breakpoint: bool = false) 🔗

Starts a debug break in script execution, optionally specifying whether the program can continue based on `can_continue` and whether the break was due to a breakpoint.

---

int **get_depth**() const 🔗

Experimental: This method may be changed or removed in future versions.

Returns the current debug depth.

---

int **get_lines_left**() const

Experimental: This method may be changed or removed in future versions.

Returns the number of lines that remain.

---

bool **has_capture**(name: StringName)

Returns `true` if a capture with the given name is present otherwise `false`.

---

bool **has_profiler**(name: StringName)

Returns `true` if a profiler with the given name is present otherwise `false`.

---

void **insert_breakpoint**(line: int, source: StringName)

Inserts a new breakpoint with the given `source` and `line`.

---

bool **is_active**()

Returns `true` if the debugger is active otherwise `false`.

---

bool **is_breakpoint**(line: int, source: StringName) const

Returns `true` if the given `source` and `line` represent an existing breakpoint.

---

bool **is_profiling**(name: StringName)

Returns `true` if a profiler with the given name is present and active otherwise `false`.

---

bool **is_skipping_breakpoints**() const

Returns `true` if the debugger is skipping breakpoints otherwise `false`.

---

void **line_poll**()

Forces a processing loop of debugger events. The purpose of this method is just processing events every now and then when the script might get too busy, so that bugs like infinite loops can be caught

---

void **profiler_add_frame_data**(name: StringName, data: Array)

Calls the `add` callable of the profiler with given `name` and `data`.

---

void **profiler_enable**(name: StringName, enable: bool, arguments: Array = [])

Calls the `toggle` callable of the profiler with given `name` and `arguments`. Enables/Disables the same profiler depending on `enable` argument.

---

void **register_message_capture**(name: StringName, callable: Callable) 🔗

Registers a message capture with given `name`. If `name` is "my_message" then messages starting with "my_message:" will be called with the given callable.

Callable must accept a message string and a data array as argument. If the message and data are valid then callable must return `true` otherwise `false`.

---

void **register_profiler**(name: StringName, profiler: EngineProfiler) 🔗

Registers a profiler with the given `name`. See EngineProfiler for more information.

---

void **remove_breakpoint**(line: int, source: StringName) 🔗

Removes a breakpoint with the given `source` and `line`.

---

void **script_debug**(language: ScriptLanguage, can_continue: bool = true, is_error_breakpoint: bool = false) 🔗

Starts a debug break in script execution, optionally specifying whether the program can continue based on `can_continue` and whether the break was due to a breakpoint.

---

void **send_message**(message: String, data: Array) 🔗

Sends a message with given `message` and `data` array.

---

void **set_depth**(depth: int)

Experimental: This method may be changed or removed in future versions.

Sets the current debugging depth.

---

void **set_lines_left**(lines: int)

Experimental: This method may be changed or removed in future versions.

Sets the current debugging lines that remain.

---

void **unregister_message_capture**(name: StringName)

Unregisters the message capture with given `name`.

---

void **unregister_profiler**(name: StringName)

Unregisters a profiler with given `name`.

EngineProfiler

Inherits: RefCounted **<** Object

Base class for creating custom profilers.

Description

This class can be used to implement custom profilers that are able to interact with the engine and editor debugger.

See EngineDebugger and EditorDebuggerPlugin for more information.

Methods

void	_add_frame(data: Array) virtual
void	_tick(frame_time: float, process_time: float, physics_time: float, physics_frame_time: float) virtual
void	_toggle(enable: bool, options: Array) virtual

Method Descriptions

void **_add_frame**(data: Array) virtual 🔗

Called when data is added to profiler using EngineDebugger.profiler_add_frame_data.

void **_tick**(frame_time: float, process_time: float, physics_time: float, physics_frame_time: float) virtual 🔗

Called once every engine iteration when the profiler is active with information about the current frame. All time values are in seconds. Lower values represent faster processing times and are therefore considered better.

void **_toggle**(enable: bool, options: Array) virtual 🔗

Called when the profiler is enabled/disabled, along with a set of `options`.

Expression

Inherits: RefCounted **<** Object

A class that stores an expression you can execute.

Description

An expression can be made of any arithmetic operation, built-in math function call, method call of a passed instance, or built-in type construction call.

An example expression text using the built-in math functions could be `sqrt(pow(3, 2) + pow(4, 2))`.

In the following example we use a LineEdit node to write our expression and show the result.

GDScript

```
var expression = Expression.new()

func _ready():
    $LineEdit.text_submitted.connect(self._on_text_submitted)

func _on_text_submitted(command):
    var error = expression.parse(command)
    if error != OK:
        print(expression.get_error_text())
        return
    var result = expression.execute()
    if not expression.has_execute_failed():
        $LineEdit.text = str(result)
```

C#

```
private Expression _expression = new Expression();

public override void _Ready()
{
    GetNode<LineEdit>("LineEdit").TextSubmitted +=
OnTextEntered;
}

private void OnTextEntered(string command)
{
    Error error = _expression.Parse(command);
    if (error != Error.Ok)
    {
        GD.Print(_expression.GetErrorText());
        return;
    }
    Variant result = _expression.Execute();
    if (!_expression.HasExecuteFailed())
    {
        GetNode<LineEdit>("LineEdit").Text = result.ToString();
    }
}
```

Tutorials

- Evaluating Expressions

Methods

Variant	execute(inputs: Array = [], base_instance: Object = null, show_error: bool = true, const_calls_only: bool = false)
String	get_error_text() const
bool	has_execute_failed() const

Error	parse(expression: String, input_names: PackedStringArray = PackedStringArray())

Method Descriptions

Variant **execute**(inputs: Array = [], base_instance: Object = null, show_error: bool = true, const_calls_only: bool = false) 🔗

Executes the expression that was previously parsed by parse and returns the result. Before you use the returned object, you should check if the method failed by calling has_execute_failed.

If you defined input variables in parse, you can specify their values in the inputs array, in the same order.

String **get_error_text**() const 🔗

Returns the error text if parse or execute has failed.

---

bool **has_execute_failed**() const 🔗

Returns `true` if execute has failed.

Error **parse**(expression: String, input_names: PackedStringArray = PackedStringArray()) 🔗

Parses the expression and returns an Error code.

You can optionally specify names of variables that may appear in the expression with `input_names`, so that you can bind them when it gets executed.

FileAccess

Inherits: RefCounted **<** Object

Provides methods for file reading and writing operations.

Description

This class can be used to permanently store data in the user device's file system and to read from it. This is useful for store game save data or player configuration files.

Here's a sample on how to write and read from a file:

GDScript

```
func save_to_file(content):
    var file = FileAccess.open("user://save_game.dat",
FileAccess.WRITE)
    file.store_string(content)

func load_from_file():
    var file = FileAccess.open("user://save_game.dat",
FileAccess.READ)
    var content = file.get_as_text()
    return content
```

C#

```
public void SaveToFile(string content)
{
    using var file = FileAccess.Open("user://save_game.dat",
FileAccess.ModeFlags.Write);
    file.StoreString(content);
}
```

```
public string LoadFromFile()
{
    using var file = FileAccess.Open("user://save_game.dat",
FileAccess.ModeFlags.Read);
    string content = file.GetAsText();
    return content;
}
```

In the example above, the file will be saved in the user data folder as specified in the Data paths documentation.

FileAccess will close when it's freed, which happens when it goes out of scope or when it gets assigned with `null`. close can be used to close it before then explicitly. In C# the reference must be disposed manually, which can be done with the `using` statement or by calling the `Dispose` method directly.

Note: To access project resources once exported, it is recommended to use ResourceLoader instead of **FileAccess**, as some files are converted to engine-specific formats and their original source files might not be present in the exported PCK package.

Note: Files are automatically closed only if the process exits "normally" (such as by clicking the window manager's close button or pressing **Alt + F4**). If you stop the project execution by pressing **F8** while the project is running, the file won't be closed as the game process will be killed. You can work around this by calling flush at regular intervals.

Tutorials

- File system
- Runtime file loading and saving
- 3D Voxel Demo [https://godotengine.org/asset-library/asset/2755]

Properties

bool	big_endian

Methods

void	close()
bool	eof_reached() const
bool	file_exists(path: String) static
void	flush()
int	get_8() const
int	get_16() const
int	get_32() const
int	get_64() const
String	get_as_text(skip_cr: bool = false) const

PackedByteArray	get_buffer(length: int) const
PackedStringArray	get_csv_line(delim: String = ",") const
float	get_double() const
Error	get_error() const
PackedByteArray	get_file_as_bytes(path: String) static
String	get_file_as_string(path: String) static
float	get_float() const
bool	get_hidden_attribute(file: String) static
int	get_length() const
String	get_line() const
String	get_md5(path: String) static
int	get_modified_time(file: String) static

Error	get_open_error() static
String	get_pascal_string()
String	get_path() const
String	get_path_absolute() const
int	get_position() const
bool	get_read_only_attribute(file: String) static
float	get_real() const
String	get_sha256(path: String) static
BitField[UnixPermissionFlags]	get_unix_permissions(file: String) static
Variant	get_var(allow_objects: bool = false) const
bool	is_open() const

FileAccess	open(path: String, flags: ModeFlags) static
FileAccess	open_compressed(path: String, mode_flags: ModeFlags, compression_mode: CompressionMode = 0) static
FileAccess	open_encrypted(path: String, mode_flags: ModeFlags, key: PackedByteArray) static
FileAccess	open_encrypted_with_pass(path: String, mode_flags: ModeFlags, pass: String) static
Error	resize(length: int)
void	seek(position: int)
void	seek_end(position: int = 0)
Error	set_hidden_attribute(file: String, hidden: bool) static
Error	set_read_only_attribute(file: String, ro: bool) static

Error	set_unix_permissions(file: String, permissions: BitField[UnixPermissionFlags]) static
void	store_8(value: int)
void	store_16(value: int)
void	store_32(value: int)
void	store_64(value: int)
void	store_buffer(buffer: PackedByteArray)
void	store_csv_line(values: PackedStringArray, delim: String = ",")
void	store_double(value: float)
void	store_float(value: float)
void	store_line(line: String)
void	store_pascal_string(string: String)

void	store_real(value: float)
void	store_string(string: String)
void	store_var(value: Variant, full_objects: bool = false)

Enumerations

enum **ModeFlags**:

ModeFlags **READ** = `1`

Opens the file for read operations. The cursor is positioned at the beginning of the file.

ModeFlags **WRITE** = `2`

Opens the file for write operations. The file is created if it does not exist, and truncated if it does.

Note: When creating a file it must be in an already existing directory. To recursively create directories for a file path, see DirAccess.make_dir_recursive.

ModeFlags **READ_WRITE** = `3`

Opens the file for read and write operations. Does not truncate the file. The cursor is positioned at the beginning of the file.

ModeFlags **WRITE_READ** = `7`

Opens the file for read and write operations. The file is created if it does not exist, and truncated if it does. The cursor is positioned at the beginning of the file.

Note: When creating a file it must be in an already existing directory. To recursively create directories for a file path, see DirAccess.make_dir_recursive.

---

enum **CompressionMode**: 🔗

CompressionMode **COMPRESSION_FASTLZ** = `0`

Uses the FastLZ [https://fastlz.org/] compression method.

CompressionMode **COMPRESSION_DEFLATE** = `1`

Uses the DEFLATE [https://en.wikipedia.org/wiki/DEFLATE] compression method.

CompressionMode **COMPRESSION_ZSTD** = `2`

Uses the Zstandard [https://facebook.github.io/zstd/] compression method.

CompressionMode **COMPRESSION_GZIP** = `3`

Uses the gzip [https://www.gzip.org/] compression method.

CompressionMode **COMPRESSION_BROTLI** = `4`

Uses the brotli [https://github.com/google/brotli] compression method (only decompression is supported).

---

flags **UnixPermissionFlags**: 🔗

UnixPermissionFlags **UNIX_READ_OWNER** = `256`

Read for owner bit.

UnixPermissionFlags **UNIX_WRITE_OWNER** = `128`

Write for owner bit.

UnixPermissionFlags **UNIX_EXECUTE_OWNER** = `64`

Execute for owner bit.

UnixPermissionFlags **UNIX_READ_GROUP** = `32`

Read for group bit.

UnixPermissionFlags **UNIX_WRITE_GROUP** = `16`

Write for group bit.

UnixPermissionFlags **UNIX_EXECUTE_GROUP** = `8`

Execute for group bit.

UnixPermissionFlags **UNIX_READ_OTHER** = `4`

Read for other bit.

UnixPermissionFlags **UNIX_WRITE_OTHER** = `2`

Write for other bit.

UnixPermissionFlags **UNIX_EXECUTE_OTHER** = `1`

Execute for other bit.

UnixPermissionFlags **UNIX_SET_USER_ID** = `2048`

Set user id on execution bit.

UnixPermissionFlags **UNIX_SET_GROUP_ID** = `1024`

Set group id on execution bit.

UnixPermissionFlags **UNIX_RESTRICTED_DELETE** = `512`

Restricted deletion (sticky) bit.

Property Descriptions

bool **big_endian**

- void **set_big_endian**(value: bool)
- bool **is_big_endian**()

If `true`, the file is read with big-endian endianness [https://en.wikipedia.org/wiki/Endianness]. If `false`, the file is read with little-endian endianness. If in doubt, leave this to `false` as most files are written with little-endian endianness.

Note: big_endian is only about the file format, not the CPU type. The CPU endianness doesn't affect the default endianness for files written.

Note: This is always reset to `false` whenever you open the file. Therefore, you must set big_endian *after* opening the file, not before.

Method Descriptions

void **close**()

Closes the currently opened file and prevents subsequent read/write operations. Use flush to persist the data to disk without closing the file.

Note: FileAccess will automatically close when it's freed, which happens when it goes out of scope or when it gets assigned with `null`. In C# the reference must be disposed after we are done using it, this can be done with the `using` statement or calling the `Dispose` method directly.

---

bool **eof_reached**() const 🔗

Returns `true` if the file cursor has already read past the end of the file.

Note: `eof_reached() == false` cannot be used to check whether there is more data available. To loop while there is more data available, use:

GDScript

```
while file.get_position() < file.get_length():
    # Read data
```

C#

```
while (file.GetPosition() < file.GetLength())
{
    // Read data
}
```

---

bool **file_exists**(path: String) static 🔗

Returns `true` if the file exists in the given path.

Note: Many resources types are imported (e.g. textures or sound files), and their source asset will not be included in the exported game, as only the imported version is used. See ResourceLoader.exists for an alternative approach that takes resource remapping into account.

For a non-static, relative equivalent, use DirAccess.file_exists.

void **flush**()

Writes the file's buffer to disk. Flushing is automatically performed when the file is closed. This means you don't need to call flush manually before closing a file. Still, calling flush can be used to ensure the data is safe even if the project crashes instead of being closed gracefully.

Note: Only call flush when you actually need it. Otherwise, it will decrease performance due to constant disk writes.

int **get_8**() const

Returns the next 8 bits from the file as an integer. See store_8 for details on what values can be stored and retrieved this way.

int **get_16**() const

Returns the next 16 bits from the file as an integer. See store_16 for details on what values can be stored and retrieved this way.

int **get_32**() const

Returns the next 32 bits from the file as an integer. See store_32 for details on what values can be stored and retrieved this way.

int **get_64**() const

Returns the next 64 bits from the file as an integer. See store_64 for details on what values can be stored and retrieved this way.

String **get_as_text**(skip_cr: bool = false) const

Returns the whole file as a String. Text is interpreted as being UTF-8 encoded.

If `skip_cr` is `true`, carriage return characters (`\r`, CR) will be ignored when parsing the UTF-8, so that only line feed characters (`\n`, LF) represent a new line (Unix convention).

PackedByteArray **get_buffer**(length: int) const

Returns next `length` bytes of the file as a PackedByteArray.

PackedStringArray **get_csv_line**(delim: String = ",") const

Returns the next value of the file in CSV (Comma-Separated Values) format. You can pass a different delimiter `delim` to use other than the default `","` (comma). This delimiter must be one-character long, and cannot be a double quotation mark.

Text is interpreted as being UTF-8 encoded. Text values must be enclosed in double quotes if they include the delimiter character. Double quotes within a text value can be escaped by doubling their occurrence.

For example, the following CSV lines are valid and will be properly parsed as two strings each:

```
Alice,"Hello, Bob!"
Bob,Alice! What a surprise!
Alice,"I thought you'd reply with ""Hello, world""."
```

Note how the second line can omit the enclosing quotes as it does not include the delimiter. However it *could* very well use quotes, it was only written without for demonstration purposes. The third line must use `""` for each quotation mark that needs to be interpreted as such instead of the end of a text value.

---

float **get_double**() const 🔗

Returns the next 64 bits from the file as a floating-point number.

Error **get_error**() const 🔗

Returns the last error that happened when trying to perform operations. Compare with the `ERR_FILE_*` constants from Error.

PackedByteArray **get_file_as_bytes**(path: String) static 🔗

Returns the whole `path` file contents as a PackedByteArray without any decoding.

Returns an empty PackedByteArray if an error occurred while opening the file. You can use get_open_error to check the error that occurred.

String **get_file_as_string**(path: String) static 🔗

Returns the whole `path` file contents as a String. Text is interpreted as being UTF-8 encoded.

Returns an empty String if an error occurred while opening the file. You can use get_open_error to check the error that occurred.

---

float **get_float**() const

Returns the next 32 bits from the file as a floating-point number.

---

bool **get_hidden_attribute**(file: String) static

Returns `true`, if file `hidden` attribute is set.

Note: This method is implemented on iOS, BSD, macOS, and Windows.

---

int **get_length**() const

Returns the size of the file in bytes.

String **get_line**() const

Returns the next line of the file as a String. The returned string doesn't include newline (`\n`) or carriage return (`\r`) characters, but does include any other leading or trailing whitespace.

Text is interpreted as being UTF-8 encoded.

String **get_md5**(path: String) static

Returns an MD5 String representing the file at the given path or an empty String on failure.

---

int **get_modified_time**(file: String) static 🔗

Returns the last time the `file` was modified in Unix timestamp format, or `0` on error. This Unix timestamp can be converted to another format using the Time singleton.

Error **get_open_error**() static 🔗

Returns the result of the last open call in the current thread.

String **get_pascal_string**() 🔗

Returns a String saved in Pascal format from the file.

Text is interpreted as being UTF-8 encoded.

String **get_path**() const 🔗

Returns the path as a String for the current open file.

String **get_path_absolute**() const 🔗

Returns the absolute path as a String for the current open file.

---

int **get_position**() const 🔗

Returns the file cursor's position.

---

bool **get_read_only_attribute**(file: String) static 🔗

Returns `true`, if file `read only` attribute is set.

Note: This method is implemented on iOS, BSD, macOS, and Windows.

---

float **get_real**() const

Returns the next bits from the file as a floating-point number.

String **get_sha256**(path: String) static

Returns an SHA-256 String representing the file at the given path or an empty String on failure.

BitField[UnixPermissionFlags] **get_unix_permissions**(file: String) static

Returns file UNIX permissions.

Note: This method is implemented on iOS, Linux/BSD, and macOS.

Variant **get_var**(allow_objects: bool = false) const

Returns the next Variant value from the file. If `allow_objects` is `true`, decoding objects is allowed.

Internally, this uses the same decoding mechanism as the @GlobalScope.bytes_to_var method.

Warning: Deserialized objects can contain code which gets executed. Do not use this option if the serialized object comes from untrusted sources to avoid potential security threats such as remote code execution.

---

bool **is_open**() const 🔗

Returns `true` if the file is currently opened.

FileAccess **open**(path: String, flags: ModeFlags) static 🔗

Creates a new **FileAccess** object and opens the file for writing or reading, depending on the flags.

Returns `null` if opening the file failed. You can use get_open_error to check the error that occurred.

FileAccess **open_compressed**(path: String, mode_flags: ModeFlags, compression_mode: CompressionMode = 0) static 🔗

Creates a new **FileAccess** object and opens a compressed file for reading or writing.

Note: open_compressed can only read files that were saved by Godot, not third-party compression formats. See GitHub issue #28999 [https://github.com/godotengine/godot/issues/28999] for a workaround.

Returns `null` if opening the file failed. You can use get_open_error to check the error that occurred.

FileAccess **open_encrypted**(path: String, mode_flags: ModeFlags, key: PackedByteArray) static 🔗

Creates a new **FileAccess** object and opens an encrypted file in write or read mode. You need to pass a binary key to encrypt/decrypt it.

Note: The provided key must be 32 bytes long.

Returns `null` if opening the file failed. You can use get_open_error to check the error that occurred.

FileAccess **open_encrypted_with_pass**(path: String, mode_flags: ModeFlags, pass: String) static

Creates a new **FileAccess** object and opens an encrypted file in write or read mode. You need to pass a password to encrypt/decrypt it.

Returns `null` if opening the file failed. You can use get_open_error to check the error that occurred.

Error **resize**(length: int)

Resizes the file to a specified length. The file must be open in a mode that permits writing. If the file is extended, NUL characters are appended. If the file is truncated, all data from the end file to the original length of the file is lost.

---

void **seek**(position: int)

Changes the file reading/writing cursor to the specified position (in bytes from the beginning of the file).

---

void **seek_end**(position: int = 0)

Changes the file reading/writing cursor to the specified position (in bytes from the end of the file).

Note: This is an offset, so you should use negative numbers or the cursor will be at the end of the file.

Error **set_hidden_attribute**(file: String, hidden: bool) static

Sets file **hidden** attribute.

Note: This method is implemented on iOS, BSD, macOS, and Windows.

Error **set_read_only_attribute**(file: String, ro: bool) static

Sets file **read only** attribute.

Note: This method is implemented on iOS, BSD, macOS, and Windows.

Error **set_unix_permissions**(file: String, permissions: BitField[UnixPermissionFlags]) static

Sets file UNIX permissions.

Note: This method is implemented on iOS, Linux/BSD, and macOS.

---

void **store_8**(value: int)

Stores an integer as 8 bits in the file.

Note: The `value` should lie in the interval `[0, 255]`. Any other value will overflow and wrap around.

To store a signed integer, use store_64, or convert it manually (see store_16 for an example).

---

void **store_16**(value: int)

Stores an integer as 16 bits in the file.

Note: The `value` should lie in the interval `[0, 2^16 - 1]`. Any other value will overflow and wrap around.

To store a signed integer, use store_64 or store a signed integer from the interval `[-2^15, 2^15 - 1]` (i.e. keeping one bit for the signedness) and compute its sign manually when reading. For example:

GDScript

```
const MAX_15B = 1 << 15
const MAX_16B = 1 << 16

func unsigned16_to_signed(unsigned):
    return (unsigned + MAX_15B) % MAX_16B - MAX_15B

func _ready():
    var f = FileAccess.open("user://file.dat", FileAccess.WRITE_READ)
    f.store_16(-42) # This wraps around and stores 65494 (2^16 - 42).
    f.store_16(121) # In bounds, will store 121.
    f.seek(0) # Go back to start to read the stored value.
    var read1 = f.get_16() # 65494
    var read2 = f.get_16() # 121
    var converted1 = unsigned16_to_signed(read1) # -42
    var converted2 = unsigned16_to_signed(read2) # 121
```

C#

```
public override void _Ready()
{
    using var f = FileAccess.Open("user://file.dat", FileAccess.ModeFlags.WriteRead);
    f.Store16(unchecked((ushort)-42)); // This wraps around and stores 65494 (2^16 - 42).
    f.Store16(121); // In bounds, will store 121.
    f.Seek(0); // Go back to start to read the stored value.
    ushort read1 = f.Get16(); // 65494
    ushort read2 = f.Get16(); // 121
    short converted1 = (short)read1; // -42
    short converted2 = (short)read2; // 121
}
```

void **store_32**(value: int)

Stores an integer as 32 bits in the file.

Note: The `value` should lie in the interval `[0, 2^32 - 1]`. Any other value will overflow and wrap around.

To store a signed integer, use store_64, or convert it manually (see store_16 for an example).

void **store_64**(value: int)

Stores an integer as 64 bits in the file.

Note: The `value` must lie in the interval `[-2^63, 2^63 - 1]` (i.e. be a valid int value).

void **store_buffer**(buffer: PackedByteArray)

Stores the given array of bytes in the file.

void **store_csv_line**(values: PackedStringArray, delim: String = ",")

Store the given PackedStringArray in the file as a line formatted in the CSV (Comma-Separated Values) format. You can pass a different delimiter `delim` to use other than the default `","` (comma). This delimiter must be one-character long.

Text will be encoded as UTF-8.

---

void **store_double**(value: float)

Stores a floating-point number as 64 bits in the file.

---

void **store_float**(value: float)

Stores a floating-point number as 32 bits in the file.

---

void **store_line**(line: String)

Appends `line` to the file followed by a line return character (`\n`), encoding the text as UTF-8.

---

void **store_pascal_string**(string: String)

Stores the given String as a line in the file in Pascal format (i.e. also store the length of the string).

Text will be encoded as UTF-8.

---

void **store_real**(value: float)

Stores a floating-point number in the file.

---

void **store_string**(string: String)

Appends `string` to the file without a line return, encoding the text as UTF-8.

Note: This method is intended to be used to write text files. The string is stored as a UTF-8 encoded buffer without string length or terminating zero, which means that it can't be loaded back easily. If you want to store a retrievable string in a binary file, consider using store_pascal_string instead. For retrieving strings from a text file, you can use `get_buffer(length).get_string_from_utf8()` (if you know the length) or get_as_text.

---

void **store_var**(value: Variant, full_objects: bool = false) 🔗

Stores any Variant value in the file. If `full_objects` is `true`, encoding objects is allowed (and can potentially include code).

Internally, this uses the same encoding mechanism as the @GlobalScope.var_to_bytes method.

Note: Not all properties are included. Only properties that are configured with the @GlobalScope.PROPERTY_USAGE_STORAGE flag set will be serialized. You can add a new usage flag to a property by overriding the Object._get_property_list method in your class. You can also check how property usage is configured by calling Object._get_property_list. See PropertyUsageFlags for the possible usage flags.

FramebufferCacheRD

Inherits: Object

Framebuffer cache manager for Rendering Device based renderers.

Description

Framebuffer cache manager for Rendering Device based renderers. Provides a way to create a framebuffer and reuse it in subsequent calls for as long as the used textures exists. Framebuffers will automatically be cleaned up when dependent objects are freed.

Methods

RID	get_cache_multipass(textures: Array[RID], passes: Array[RDFramebufferPass], views: int) static

---

Method Descriptions

RID **get_cache_multipass**(textures: Array[RID], passes: Array[RDFramebufferPass], views: int) static 🔗

Creates, or obtains a cached, framebuffer. `textures` lists textures accessed. `passes` defines the subpasses and texture allocation, if left empty a single pass is created and textures are allocated depending on their usage flags. `views` defines the number of views used when rendering.

GDExtensionManager

Inherits: Object

Provides access to GDExtension functionality.

Description

The GDExtensionManager loads, initializes, and keeps track of all available GDExtension libraries in the project.

Note: Do not worry about GDExtension unless you know what you are doing.

Tutorials

- GDExtension overview
- GDExtension example in C++

Methods

GDExtension	get_extension(path: String)
PackedStringArray	get_loaded_extensions() const
bool	is_extension_loaded(path: String) const

LoadStatus	load_extension(path: String)
LoadStatus	reload_extension(path: String)
LoadStatus	unload_extension(path: String)

Signals

extensions_reloaded()

Emitted after the editor has finished reloading one or more extensions.

Enumerations

enum **LoadStatus**:

LoadStatus **LOAD_STATUS_OK** = `0`

The extension has loaded successfully.

LoadStatus **LOAD_STATUS_FAILED** = `1`

The extension has failed to load, possibly because it does not exist or has missing dependencies.

LoadStatus **LOAD_STATUS_ALREADY_LOADED** = `2`

The extension has already been loaded.

LoadStatus **LOAD_STATUS_NOT_LOADED** = `3`

The extension has not been loaded.

LoadStatus **LOAD_STATUS_NEEDS_RESTART** = `4`

The extension requires the application to restart to fully load.

Method Descriptions

GDExtension **get_extension**(path: String) 🔗

Returns the GDExtension at the given file `path`, or `null` if it has not been loaded or does not exist.

PackedStringArray **get_loaded_extensions**() const 🔗

Returns the file paths of all currently loaded extensions.

---

bool **is_extension_loaded**(path: String) const 🔗

Returns `true` if the extension at the given file `path` has already been loaded successfully. See also get_loaded_extensions.

LoadStatus **load_extension**(path: String) 🔗

Loads an extension by absolute file path. The `path` needs to point to a valid GDExtension. Returns LOAD_STATUS_OK if successful.

LoadStatus **reload_extension**(path: String) 🔗

Reloads the extension at the given file path. The `path` needs to point to a valid GDExtension, otherwise this method may return either LOAD_STATUS_NOT_LOADED or LOAD_STATUS_FAILED.

Note: You can only reload extensions in the editor. In release builds, this method always fails and returns LOAD_STATUS_FAILED.

LoadStatus **unload_extension**(path: String) 🔗

Unloads an extension by file path. The `path` needs to point to an already loaded GDExtension, otherwise this method returns LOAD_STATUS_NOT_LOADED.

Geometry2D

Inherits: Object

Provides methods for some common 2D geometric operations.

Description

Provides a set of helper functions to create geometric shapes, compute intersections between shapes, and process various other geometric operations in 2D.

Methods

Array[PackedVector2Array]	clip_polygons(polygon_a: PackedVector2Array, polygon_b: PackedVector2Array)
Array[PackedVector2Array]	clip_polyline_with_polygon(polyline: PackedVector2Array, polygon: PackedVector2Array)
PackedVector2Array	convex_hull(points: PackedVector2Array)
Array[PackedVector2Array]	decompose_polygon_in_convex(polygon: PackedVector2Array)
Array[PackedVector2Array]	exclude_polygons(polygon_a: PackedVector2Array, polygon_b: PackedVector2Array)

Vector2	get_closest_point_to_segment(point: Vector2, s1: Vector2, s2: Vector2)
Vector2	get_closest_point_to_segment_uncapped(point: Vector2, s1: Vector2, s2: Vector2)
PackedVector2Array	get_closest_points_between_segments(p1: Vector2, q1: Vector2, p2: Vector2, q2: Vector2)
Array[PackedVector2Array]	intersect_polygons(polygon_a: PackedVector2Array, polygon_b: PackedVector2Array)
Array[PackedVector2Array]	intersect_polyline_with_polygon(polyline: PackedVector2Array, polygon: PackedVector2Array)
bool	is_point_in_circle(point: Vector2, circle_position: Vector2, circle_radius: float)
bool	is_point_in_polygon(point: Vector2, polygon: PackedVector2Array)
bool	is_polygon_clockwise(polygon: PackedVector2Array)
Variant	line_intersects_line(from_a: Vector2, dir_a: Vector2, from_b: Vector2, dir_b: Vector2)
Dictionary	make_atlas(sizes: PackedVector2Array)

Array[PackedVector2Array]	merge_polygons(polygon_a: PackedVector2Array, polygon_b: PackedVector2Array)
Array[PackedVector2Array]	offset_polygon(polygon: PackedVector2Array, delta: float, join_type: PolyJoinType = 0)
Array[PackedVector2Array]	offset_polyline(polyline: PackedVector2Array, delta: float, join_type: PolyJoinType = 0, end_type: PolyEndType = 3)
bool	point_is_inside_triangle(point: Vector2, a: Vector2, b: Vector2, c: Vector2) const
float	segment_intersects_circle(segment_from: Vector2, segment_to: Vector2, circle_position: Vector2, circle_radius: float)
Variant	segment_intersects_segment(from_a: Vector2, to_a: Vector2, from_b: Vector2, to_b: Vector2)
PackedInt32Array	triangulate_delaunay(points: PackedVector2Array)
PackedInt32Array	triangulate_polygon(polygon: PackedVector2Array)

Enumerations

enum **PolyBooleanOperation**: 🔗

PolyBooleanOperation **OPERATION_UNION** = `0`

Create regions where either subject or clip polygons (or both) are filled.

PolyBooleanOperation **OPERATION_DIFFERENCE** = `1`

Create regions where subject polygons are filled except where clip polygons are filled.

PolyBooleanOperation **OPERATION_INTERSECTION** = `2`

Create regions where both subject and clip polygons are filled.

PolyBooleanOperation **OPERATION_XOR** = `3`

Create regions where either subject or clip polygons are filled but not where both are filled.

enum **PolyJoinType**: 🔗

PolyJoinType **JOIN_SQUARE** = `0`

Squaring is applied uniformally at all convex edge joins at `1 * delta`.

PolyJoinType **JOIN_ROUND** = `1`

While flattened paths can never perfectly trace an arc, they are approximated by a series of arc chords.

PolyJoinType **JOIN_MITER** = `2`

There's a necessary limit to mitered joins since offsetting edges that join at very acute angles will produce excessively long and narrow "spikes". For any given edge join, when miter offsetting would exceed that maximum distance, "square" joining is applied.

enum **PolyEndType**: 🔗

PolyEndType **END_POLYGON** = `0`

Endpoints are joined using the PolyJoinType value and the path filled as a polygon.

PolyEndType **END_JOINED** = `1`

Endpoints are joined using the PolyJoinType value and the path filled as a polyline.

PolyEndType **END_BUTT** = `2`

Endpoints are squared off with no extension.

PolyEndType **END_SQUARE** = `3`

Endpoints are squared off and extended by `delta` units.

PolyEndType **END_ROUND** = `4`

Endpoints are rounded off and extended by `delta` units.

Method Descriptions

Array[PackedVector2Array] **clip_polygons**(polygon_a: PackedVector2Array, polygon_b: PackedVector2Array)

Clips `polygon_a` against `polygon_b` and returns an array of clipped polygons. This performs OPERATION_DIFFERENCE between polygons. Returns an empty array if `polygon_b` completely overlaps `polygon_a`.

If `polygon_b` is enclosed by `polygon_a`, returns an outer polygon (boundary) and inner polygon (hole) which could be distinguished by calling is_polygon_clockwise.

Array[PackedVector2Array] **clip_polyline_with_polygon**(polyline: PackedVector2Array, polygon: PackedVector2Array)

Clips `polyline` against `polygon` and returns an array of clipped polylines. This performs OPERATION_DIFFERENCE between the polyline and the polygon.

This operation can be thought of as cutting a line with a closed shape.

PackedVector2Array **convex_hull**(points: PackedVector2Array)

Given an array of Vector2s, returns the convex hull as a list of points in counterclockwise order. The last point is the same as the first one.

Array[PackedVector2Array] **decompose_polygon_in_convex**(polygon: PackedVector2Array)

Decomposes the `polygon` into multiple convex hulls and returns an array of PackedVector2Array.

Array[PackedVector2Array] **exclude_polygons**(polygon_a: PackedVector2Array, polygon_b: PackedVector2Array)

Mutually excludes common area defined by intersection of `polygon_a` and `polygon_b` (see intersect_polygons) and returns an array of excluded polygons. This performs OPERATION_XOR between polygons. In other words, returns all but common area between polygons.

The operation may result in an outer polygon (boundary) and inner polygon (hole) produced which could be distinguished by calling is_polygon_clockwise.

Vector2 **get_closest_point_to_segment**(point: Vector2, s1: Vector2, s2: Vector2)

Returns the 2D point on the 2D segment (`s1`, `s2`) that is closest to `point`. The returned point will always be inside the specified segment.

Vector2 **get_closest_point_to_segment_uncapped**(point: Vector2, s1: Vector2, s2: Vector2)

Returns the 2D point on the 2D line defined by (`s1`, `s2`) that is closest to `point`. The returned point can be inside the segment (`s1`, `s2`) or outside of it, i.e. somewhere on the line extending from the segment.

PackedVector2Array **get_closest_points_between_segments**(p1: Vector2, q1: Vector2, p2: Vector2, q2: Vector2) 🔗

Given the two 2D segments (`p1`, `q1`) and (`p2`, `q2`), finds those two points on the two segments that are closest to each other. Returns a PackedVector2Array that contains this point on (`p1`, `q1`) as well the accompanying point on (`p2`, `q2`).

Array[PackedVector2Array] **intersect_polygons**(polygon_a: PackedVector2Array, polygon_b: PackedVector2Array) 🔗

Intersects `polygon_a` with `polygon_b` and returns an array of intersected polygons. This performs OPERATION_INTERSECTION between polygons. In other words, returns common area shared by polygons. Returns an empty array if no intersection occurs.

The operation may result in an outer polygon (boundary) and inner polygon (hole) produced which could be distinguished by calling is_polygon_clockwise.

Array[PackedVector2Array] **intersect_polyline_with_polygon**(polyline: PackedVector2Array, polygon: PackedVector2Array) 🔗

Intersects `polyline` with `polygon` and returns an array of intersected polylines. This performs OPERATION_INTERSECTION between the polyline and the polygon. This operation can be thought of as chopping a line with a closed shape.

---

bool **is_point_in_circle**(point: Vector2, circle_position: Vector2, circle_radius: float) 🔗

Returns `true` if `point` is inside the circle or if it's located exactly *on* the circle's boundary, otherwise returns `false`.

---

bool **is_point_in_polygon**(point: Vector2, polygon: PackedVector2Array)

Returns `true` if `point` is inside `polygon` or if it's located exactly *on* polygon's boundary, otherwise returns `false`.

---

bool **is_polygon_clockwise**(polygon: PackedVector2Array)

Returns `true` if `polygon`'s vertices are ordered in clockwise order, otherwise returns `false`.

Note: Assumes a Cartesian coordinate system where `+x` is right and `+y` is up. If using screen coordinates (`+y` is down), the result will need to be flipped (i.e. a `true` result will indicate counter-clockwise).

Variant **line_intersects_line**(from_a: Vector2, dir_a: Vector2, from_b: Vector2, dir_b: Vector2)

Checks if the two lines (`from_a`, `dir_a`) and (`from_b`, `dir_b`) intersect. If yes, return the point of intersection as Vector2. If no intersection takes place, returns `null`.

Note: The lines are specified using direction vectors, not end points.

Dictionary **make_atlas**(sizes: PackedVector2Array)

Given an array of Vector2s representing tiles, builds an atlas. The returned dictionary has two keys: `points` is a PackedVector2Array that specifies the positions of each tile, `size` contains the overall size of the whole atlas as Vector2i.

Array[PackedVector2Array] **merge_polygons**(polygon_a: PackedVector2Array, polygon_b: PackedVector2Array)

Merges (combines) `polygon_a` and `polygon_b` and returns an array of merged polygons. This performs OPERATION_UNION between polygons.

The operation may result in an outer polygon (boundary) and multiple inner polygons (holes) produced which could be distinguished by calling is_polygon_clockwise.

Array[PackedVector2Array] **offset_polygon**(polygon: PackedVector2Array, delta: float, join_type: PolyJoinType = 0) 🔗

Inflates or deflates `polygon` by `delta` units (pixels). If `delta` is positive, makes the polygon grow outward. If `delta` is negative, shrinks the polygon inward. Returns an array of polygons because inflating/deflating may result in multiple discrete polygons. Returns an empty array if `delta` is negative and the absolute value of it approximately exceeds the minimum bounding rectangle dimensions of the polygon.

Each polygon's vertices will be rounded as determined by `join_type`, see PolyJoinType.

The operation may result in an outer polygon (boundary) and inner polygon (hole) produced which could be distinguished by calling is_polygon_clockwise.

Note: To translate the polygon's vertices specifically, multiply them to a Transform2D:

GDScript

```
var polygon = PackedVector2Array([Vector2(0, 0), Vector2(100, 0),
Vector2(100, 100), Vector2(0, 100)])
var offset = Vector2(50, 50)
polygon = Transform2D(0, offset) * polygon
print(polygon) # prints [(50, 50), (150, 50), (150, 150), (50, 150)]
```

C#

```
var polygon = new Vector2[] { new Vector2(0, 0), new Vector2(100, 0),
new Vector2(100, 100), new Vector2(0, 100) };
var offset = new Vector2(50, 50);
polygon = new Transform2D(0, offset) * polygon;
GD.Print((Variant)polygon); // prints [(50, 50), (150, 50), (150,
150), (50, 150)]
```

Array[PackedVector2Array] **offset_polyline**(polyline: PackedVector2Array, delta: float, join_type: PolyJoinType = 0, end_type: PolyEndType = 3) 🔗

Inflates or deflates `polyline` by `delta` units (pixels), producing polygons. If `delta` is positive, makes the polyline grow outward. Returns an array of polygons because inflating/deflating may result in multiple discrete polygons. If `delta` is negative, returns an empty array.

Each polygon's vertices will be rounded as determined by `join_type`, see PolyJoinType.

Each polygon's endpoints will be rounded as determined by `end_type`, see PolyEndType.

The operation may result in an outer polygon (boundary) and inner polygon (hole) produced which could be distinguished by calling is_polygon_clockwise.

---

bool **point_is_inside_triangle**(point: Vector2, a: Vector2, b: Vector2, c: Vector2) const 🔗

Returns if `point` is inside the triangle specified by `a`, `b` and `c`.

---

float **segment_intersects_circle**(segment_from: Vector2, segment_to: Vector2, circle_position: Vector2, circle_radius: float) 🔗

Given the 2D segment (`segment_from`, `segment_to`), returns the position on the segment (as a number between 0 and 1) at which the segment hits the circle that is located at position `circle_position` and has radius `circle_radius`. If the segment does not intersect the circle, -1 is returned (this is also the case if the line extending the segment would intersect the circle, but the segment does not).

Variant **segment_intersects_segment**(from_a: Vector2, to_a: Vector2, from_b: Vector2, to_b: Vector2) 🔗

Checks if the two segments (`from_a`, `to_a`) and (`from_b`, `to_b`) intersect. If yes, return the point of intersection as Vector2. If no intersection takes place, returns `null`.

PackedInt32Array **triangulate_delaunay**(points: PackedVector2Array) 🔗

Triangulates the area specified by discrete set of `points` such that no point is inside the circumcircle of any resulting triangle. Returns a PackedInt32Array where each triangle consists of three consecutive point indices into `points` (i.e. the returned array will have `n * 3` elements, with `n` being the number of found triangles). If the triangulation did not succeed, an empty PackedInt32Array is returned.

PackedInt32Array **triangulate_polygon**(polygon: PackedVector2Array) 🔗

Triangulates the polygon specified by the points in `polygon`. Returns a PackedInt32Array where each triangle consists of three consecutive point indices into `polygon` (i.e. the returned array will have `n * 3` elements, with `n` being the number of found triangles). Output triangles will always be counter clockwise, and the contour will be flipped if it's clockwise. If the triangulation did not succeed, an empty PackedInt32Array is returned.

Geometry3D

Inherits: Object

Provides methods for some common 3D geometric operations.

Description

Provides a set of helper functions to create geometric shapes, compute intersections between shapes, and process various other geometric operations in 3D.

Methods

Array[Plane]	build_box_planes(extents: Vector3)
Array[Plane]	build_capsule_planes(radius: float, height: float, sides: int, lats: int, axis: Vector3.Axis = 2)
Array[Plane]	build_cylinder_planes(radius: float, height: float, sides: int, axis: Vector3.Axis = 2)
PackedVector3Array	clip_polygon(points: PackedVector3Array, plane: Plane)
PackedVector3Array	compute_convex_mesh_points(planes: Array[Plane])

Vector3	get_closest_point_to_segment(point: Vector3, s1: Vector3, s2: Vector3)
Vector3	get_closest_point_to_segment_uncapped(point: Vector3, s1: Vector3, s2: Vector3)
PackedVector3Array	get_closest_points_between_segments(p1: Vector3, p2: Vector3, q1: Vector3, q2: Vector3)
Vector3	get_triangle_barycentric_coords(point: Vector3, a: Vector3, b: Vector3, c: Vector3)
Variant	ray_intersects_triangle(from: Vector3, dir: Vector3, a: Vector3, b: Vector3, c: Vector3)
PackedVector3Array	segment_intersects_convex(from: Vector3, to: Vector3, planes: Array[Plane])
PackedVector3Array	segment_intersects_cylinder(from: Vector3, to: Vector3, height: float, radius: float)
PackedVector3Array	segment_intersects_sphere(from: Vector3, to: Vector3, sphere_position: Vector3, sphere_radius: float)
Variant	segment_intersects_triangle(from: Vector3, to: Vector3, a: Vector3, b: Vector3, c: Vector3)

PackedInt32Array	tetrahedralize_delaunay(points: PackedVector3Array)

Method Descriptions

Array[Plane] **build_box_planes**(extents: Vector3) 🔗

Returns an array with 6 Planes that describe the sides of a box centered at the origin. The box size is defined by `extents`, which represents one (positive) corner of the box (i.e. half its actual size).

Array[Plane] **build_capsule_planes**(radius: float, height: float, sides: int, lats: int, axis: Vector3.Axis = 2) 🔗

Returns an array of Planes closely bounding a faceted capsule centered at the origin with radius `radius` and height `height`. The parameter `sides` defines how many planes will be generated for the side part of the capsule, whereas `lats` gives the number of latitudinal steps at the bottom and top of the capsule. The parameter `axis` describes the axis along which the capsule is oriented (0 for X, 1 for Y, 2 for Z).

Array[Plane] **build_cylinder_planes**(radius: float, height: float, sides: int, axis: Vector3.Axis = 2) 🔗

Returns an array of Planes closely bounding a faceted cylinder centered at the origin with radius `radius` and height `height`. The parameter `sides` defines how many planes will be generated for the round part of the cylinder. The parameter `axis` describes the axis along which the cylinder is oriented (0 for X, 1 for Y, 2 for Z).

PackedVector3Array **clip_polygon**(points: PackedVector3Array, plane: Plane)

Clips the polygon defined by the points in `points` against the `plane` and returns the points of the clipped polygon.

PackedVector3Array **compute_convex_mesh_points**(planes: Array[Plane])

Calculates and returns all the vertex points of a convex shape defined by an array of `planes`.

Vector3 **get_closest_point_to_segment**(point: Vector3, s1: Vector3, s2: Vector3)

Returns the 3D point on the 3D segment (`s1`, `s2`) that is closest to `point`. The returned point will always be inside the specified segment.

Vector3 **get_closest_point_to_segment_uncapped**(point: Vector3, s1: Vector3, s2: Vector3)

Returns the 3D point on the 3D line defined by (`s1`, `s2`) that is closest to `point`. The returned point can be inside the segment (`s1`, `s2`) or outside of it, i.e. somewhere on the line extending from the segment.

PackedVector3Array **get_closest_points_between_segments**(p1: Vector3, p2: Vector3, q1: Vector3, q2: Vector3)

Given the two 3D segments (`p1`, `p2`) and (`q1`, `q2`), finds those two points on the two segments that are closest to each other. Returns a PackedVector3Array that contains this point on (`p1`, `p2`) as well the accompanying point on (`q1`, `q2`).

Vector3 **get_triangle_barycentric_coords**(point: Vector3, a: Vector3, b: Vector3, c: Vector3)

Returns a Vector3 containing weights based on how close a 3D position (`point`) is to a triangle's different vertices (`a`, `b` and `c`). This is useful for interpolating between the data of different vertices in a triangle. One example use case is using this to smoothly rotate over a mesh instead of relying solely on face normals.

Here is a more detailed explanation of barycentric coordinates. [https://en.wikipedia.org/wiki/Barycentric_coordinate_system]

Variant **ray_intersects_triangle**(from: Vector3, dir: Vector3, a: Vector3, b: Vector3, c: Vector3)

Tests if the 3D ray starting at `from` with the direction of `dir` intersects the triangle specified by `a`, `b` and `c`. If yes, returns the point of intersection as Vector3. If no intersection takes place, returns `null`.

PackedVector3Array **segment_intersects_convex**(from: Vector3, to: Vector3, planes: Array[Plane])

Given a convex hull defined though the Planes in the array `planes`, tests if the segment (`from`, `to`) intersects with that hull. If an intersection is found, returns a PackedVector3Array containing the point the intersection and the hull's normal. Otherwise, returns an empty array.

PackedVector3Array **segment_intersects_cylinder**(from: Vector3, to: Vector3, height: float, radius: float)

Checks if the segment (`from`, `to`) intersects the cylinder with height `height` that is centered at the origin and has radius `radius`. If no, returns an empty PackedVector3Array. If an intersection takes place, the returned array contains the point of intersection and the cylinder's normal at the point of intersection.

PackedVector3Array **segment_intersects_sphere**(from: Vector3, to: Vector3, sphere_position: Vector3, sphere_radius: float)

Checks if the segment (`from`, `to`) intersects the sphere that is located at `sphere_position` and has radius `sphere_radius`. If no, returns an empty PackedVector3Array. If yes, returns a PackedVector3Array containing the point of intersection and the sphere's normal at the point of intersection.

Variant **segment_intersects_triangle**(from: Vector3, to: Vector3, a: Vector3, b: Vector3, c: Vector3)

Tests if the segment (`from`, `to`) intersects the triangle `a`, `b`, `c`. If yes, returns the point of intersection as Vector3. If no intersection takes place, returns `null`.

PackedInt32Array **tetrahedralize_delaunay**(points: PackedVector3Array)

Tetrahedralizes the volume specified by a discrete set of `points` in 3D space, ensuring that no point lies within the circumsphere of any resulting tetrahedron. The method returns a PackedInt32Array where each tetrahedron consists of four consecutive point indices into the `points` array (resulting in an array with `n * 4` elements, where `n` is the number of tetrahedra found). If the tetrahedralization is unsuccessful, an empty PackedInt32Array is returned.

HashingContext

Inherits: RefCounted **<** Object

Provides functionality for computing cryptographic hashes chunk by chunk.

Description

The HashingContext class provides an interface for computing cryptographic hashes over multiple iterations. Useful for computing hashes of big files (so you don't have to load them all in memory), network streams, and data streams in general (so you don't have to hold buffers).

The HashType enum shows the supported hashing algorithms.

GDScript

```
const CHUNK_SIZE = 1024

func hash_file(path):
    # Check that file exists.
    if not FileAccess.file_exists(path):
        return
    # Start an SHA-256 context.
    var ctx = HashingContext.new()
    ctx.start(HashingContext.HASH_SHA256)
    # Open the file to hash.
    var file = FileAccess.open(path, FileAccess.READ)
    # Update the context after reading each chunk.
    while file.get_position() < file.get_length():
        var remaining = file.get_length() - file.get_position()
        ctx.update(file.get_buffer(min(remaining, CHUNK_SIZE)))
    # Get the computed hash.
    var res = ctx.finish()
    # Print the result as hex string and array.
    printt(res.hex_encode(), Array(res))
```

C#

```
public const int ChunkSize = 1024;

public void HashFile(string path)
{
    // Check that file exists.
    if (!FileAccess.FileExists(path))
    {
        return;
    }
    // Start an SHA-256 context.
    var ctx = new HashingContext();
    ctx.Start(HashingContext.HashType.Sha256);
    // Open the file to hash.
    using var file = FileAccess.Open(path,
FileAccess.ModeFlags.Read);
    // Update the context after reading each chunk.
    while (file.GetPosition() < file.GetLength())
    {
        int remaining = (int)(file.GetLength() -
file.GetPosition());
        ctx.Update(file.GetBuffer(Mathf.Min(remaining,
ChunkSize)));
    }
    // Get the computed hash.
    byte[] res = ctx.Finish();
    // Print the result as hex string and array.
    GD.PrintT(res.HexEncode(), (Variant)res);
}
```

Methods

PackedByteArray finish()

Error	start(type: HashType)
Error	update(chunk: PackedByteArray)

Enumerations

enum **HashType**: 🔗

HashType **HASH_MD5** = `0`

Hashing algorithm: MD5.

HashType **HASH_SHA1** = `1`

Hashing algorithm: SHA-1.

HashType **HASH_SHA256** = `2`

Hashing algorithm: SHA-256.

Method Descriptions

PackedByteArray **finish**() 🔗

Closes the current context, and return the computed hash.

Error **start**(type: HashType) 🔗

Starts a new hash computation of the given `type` (e.g. HASH_SHA256 to start computation of an SHA-256).

Error **update**(chunk: PackedByteArray) 🔗

Updates the computation with the given `chunk` of data.

HMACContext

Inherits: RefCounted **<** Object

Used to create an HMAC for a message using a key.

Description

The HMACContext class is useful for advanced HMAC use cases, such as streaming the message as it supports creating the message over time rather than providing it all at once.

GDScript

```
extends Node
var ctx = HMACContext.new()

func _ready():
    var key = "supersecret".to_utf8_buffer()
    var err = ctx.start(HashingContext.HASH_SHA256, key)
    assert(err == OK)
    var msg1 = "this is ".to_utf8_buffer()
    var msg2 = "super duper secret".to_utf8_buffer()
    err = ctx.update(msg1)
    assert(err == OK)
    err = ctx.update(msg2)
    assert(err == OK)
    var hmac = ctx.finish()
    print(hmac.hex_encode())
```

C#

```
using Godot;
using System.Diagnostics;

public partial class MyNode : Node
```

```
{
    private HmacContext _ctx = new HmacContext();

    public override void _Ready()
    {
        byte[] key = "supersecret".ToUtf8Buffer();
        Error err = _ctx.Start(HashingContext.HashType.Sha256,
key);
        Debug.Assert(err == Error.Ok);
        byte[] msg1 = "this is ".ToUtf8Buffer();
        byte[] msg2 = "super duper secret".ToUtf8Buffer();
        err = _ctx.Update(msg1);
        Debug.Assert(err == Error.Ok);
        err = _ctx.Update(msg2);
        Debug.Assert(err == Error.Ok);
        byte[] hmac = _ctx.Finish();
        GD.Print(hmac.HexEncode());
    }
}
```

Methods

PackedByteArray	finish()
Error	start(hash_type: HashType, key: PackedByteArray)
Error	update(data: PackedByteArray)

Method Descriptions

PackedByteArray **finish**() 🔗

Returns the resulting HMAC. If the HMAC failed, an empty PackedByteArray is returned.

Error **start**(hash_type: HashType, key: PackedByteArray) 🔗

Initializes the HMACContext. This method cannot be called again on the same HMACContext until finish has been called.

Error **update**(data: PackedByteArray) 🔗

Updates the message to be HMACed. This can be called multiple times before finish is called to append `data` to the message, but cannot be called until start has been called.

HTTPClient

Inherits: RefCounted **<** Object

Low-level hyper-text transfer protocol client.

Description

Hyper-text transfer protocol client (sometimes called "User Agent"). Used to make HTTP requests to download web content, upload files and other data or to communicate with various services, among other use cases.

See the HTTPRequest node for a higher-level alternative.

Note: This client only needs to connect to a host once (see connect_to_host) to send multiple requests. Because of this, methods that take URLs usually take just the part after the host instead of the full URL, as the client is already connected to a host. See request for a full example and to get started.

A **HTTPClient** should be reused between multiple requests or to connect to different hosts instead of creating one client per request. Supports Transport Layer Security (TLS), including server certificate verification. HTTP status codes in the 2xx range indicate success, 3xx redirection (i.e. "try again, but over here"), 4xx something was wrong with the request, and 5xx something went wrong on the server's side.

For more information on HTTP, see MDN's documentation on HTTP [https://developer.mozilla.org/en-US/docs/Web/HTTP] (or read RFC 2616 [https://tools.ietf.org/html/rfc2616] to get it straight from the source).

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Note: It's recommended to use transport encryption (TLS) and to avoid sending sensitive information (such as login credentials) in HTTP GET URL parameters. Consider using HTTP POST requests or HTTP headers for such information instead.

Note: When performing HTTP requests from a project exported to Web, keep in mind the remote server may not allow requests from foreign origins due to CORS [https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS]. If you host the server in question, you should modify its backend to allow requests from foreign origins by adding the `Access-Control-Allow-Origin: *` HTTP header.

Note: TLS support is currently limited to TLS 1.0, TLS 1.1, and TLS 1.2. Attempting to connect to a TLS 1.3-only server will return an error.

Warning: TLS certificate revocation and certificate pinning are currently not supported. Revoked certificates are accepted as long as they are otherwise valid. If this is a concern, you may want to use automatically managed certificates with a short validity period.

Tutorials

- HTTP client class
- TLS certificates

Properties

bool	blocking_mode_enabled	`false`
StreamPeer	connection	
int	read_chunk_size	`65536`

Methods

void	close()
Error	connect_to_host(host: String, port: int = -1, tls_options: TLSOptions = null)
int	get_response_body_length() const
int	get_response_code() const
PackedStringArray	get_response_headers()
Dictionary	get_response_headers_as_dictionary()
Status	get_status() const
bool	has_response() const

bool	is_response_chunked() const
Error	poll()
String	query_string_from_dict(fields: Dictionary)
PackedByteArray	read_response_body_chunk()
Error	request(method: Method, url: String, headers: PackedStringArray, body: String = "")
Error	request_raw(method: Method, url: String, headers: PackedStringArray, body: PackedByteArray)
void	set_http_proxy(host: String, port: int)
void	set_https_proxy(host: String, port: int)

Enumerations

enum **Method**: 🔗

Method **METHOD_GET** = 0

HTTP GET method. The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

Method **METHOD_HEAD** = `1`

HTTP HEAD method. The HEAD method asks for a response identical to that of a GET request, but without the response body. This is useful to request metadata like HTTP headers or to check if a resource exists.

Method **METHOD_POST** = `2`

HTTP POST method. The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server. This is often used for forms and submitting data or uploading files.

Method **METHOD_PUT** = `3`

HTTP PUT method. The PUT method asks to replace all current representations of the target resource with the request payload. (You can think of POST as "create or update" and PUT as "update", although many services tend to not make a clear distinction or change their meaning).

Method **METHOD_DELETE** = `4`

HTTP DELETE method. The DELETE method requests to delete the specified resource.

Method **METHOD_OPTIONS** = `5`

HTTP OPTIONS method. The OPTIONS method asks for a description of the communication options for the target resource. Rarely used.

Method **METHOD_TRACE** = `6`

HTTP TRACE method. The TRACE method performs a message loop-back test along the path to the target resource. Returns the entire HTTP request received in the response body. Rarely used.

Method **METHOD_CONNECT** = `7`

HTTP CONNECT method. The CONNECT method establishes a tunnel to the server identified by the target resource. Rarely used.

Method **METHOD_PATCH** = `8`

HTTP PATCH method. The PATCH method is used to apply partial modifications to a resource.

Method **METHOD_MAX** = `9`

Represents the size of the Method enum.

enum **Status**: 🔗

Status **STATUS_DISCONNECTED** = `0`

Status: Disconnected from the server.

Status **STATUS_RESOLVING** = `1`

Status: Currently resolving the hostname for the given URL into an IP.

Status **STATUS_CANT_RESOLVE** = `2`

Status: DNS failure: Can't resolve the hostname for the given URL.

Status **STATUS_CONNECTING** = `3`

Status: Currently connecting to server.

Status **STATUS_CANT_CONNECT** = `4`

Status: Can't connect to the server.

Status **STATUS_CONNECTED** = `5`

Status: Connection established.

Status **STATUS_REQUESTING** = `6`

Status: Currently sending request.

Status **STATUS_BODY** = `7`

Status: HTTP body received.

Status **STATUS_CONNECTION_ERROR** = `8`

Status: Error in HTTP connection.

Status **STATUS_TLS_HANDSHAKE_ERROR** = `9`

Status: Error in TLS handshake.

---

enum **ResponseCode**: 🔗

ResponseCode **RESPONSE_CONTINUE** = `100`

HTTP status code `100 Continue`. Interim response that indicates everything so far is OK and that the client should continue with the request (or ignore this status if already finished).

ResponseCode **RESPONSE_SWITCHING_PROTOCOLS** = `101`

HTTP status code `101 Switching Protocol`. Sent in response to an `Upgrade` request header by the client. Indicates the protocol the server is switching to.

ResponseCode **RESPONSE_PROCESSING** = `102`

HTTP status code `102 Processing` (WebDAV). Indicates that the server has received and is processing the request, but no response is available yet.

ResponseCode **RESPONSE_OK** = `200`

HTTP status code `200 OK`. The request has succeeded. Default response for successful requests. Meaning varies depending on the request. GET: The resource has been fetched and is transmitted in the message body. HEAD: The entity headers are in the message body. POST: The resource describing the result of the action is transmitted in the message body. TRACE: The message body contains the request message as received by the server.

ResponseCode **RESPONSE_CREATED** = `201`

HTTP status code `201 Created`. The request has succeeded and a new resource has been created as a result of it. This is typically the response sent after a PUT request.

ResponseCode **RESPONSE_ACCEPTED** = `202`

HTTP status code `202 Accepted`. The request has been received but not yet acted upon. It is non-committal, meaning that there is no way in HTTP to later send an asynchronous response indicating the outcome of processing the request. It is intended for cases where another process or server handles the request, or for batch processing.

ResponseCode
RESPONSE_NON_AUTHORITATIVE_INFORMATION = `203`

HTTP status code `203 Non-Authoritative Information`. This response code means returned meta-information set is not exact set as available from the origin server, but collected from a local or a third party copy. Except this condition, 200 OK response should be preferred instead of this response.

ResponseCode **RESPONSE_NO_CONTENT** = `204`

HTTP status code `204 No Content`. There is no content to send for this request, but the headers may be useful. The user-agent may update its cached headers for this resource with the new ones.

ResponseCode **RESPONSE_RESET_CONTENT** = `205`

HTTP status code `205 Reset Content`. The server has fulfilled the request and desires that the client resets the "document view" that caused the request to be sent to its original state as received from the origin server.

ResponseCode **RESPONSE_PARTIAL_CONTENT** = `206`

HTTP status code `206 Partial Content`. This response code is used because of a range header sent by the client to separate download into multiple streams.

ResponseCode **RESPONSE_MULTI_STATUS** = `207`

HTTP status code `207 Multi-Status` (WebDAV). A Multi-Status response conveys information about multiple resources in situations where multiple status codes might be appropriate.

ResponseCode **RESPONSE_ALREADY_REPORTED** = `208`

HTTP status code `208 Already Reported` (WebDAV). Used inside a DAV: propstat response element to avoid enumerating the internal members of multiple bindings to the same collection repeatedly.

ResponseCode **RESPONSE_IM_USED** = `226`

HTTP status code `226 IM Used` (WebDAV). The server has fulfilled a GET request for the resource, and the response is a representation of the result of one or more instance-manipulations applied to the current instance.

ResponseCode **RESPONSE_MULTIPLE_CHOICES** = `300`

HTTP status code `300 Multiple Choice`. The request has more than one possible responses and there is no standardized way to choose one of the responses. User-agent or user should choose one of them.

ResponseCode **RESPONSE_MOVED_PERMANENTLY** = `301`

HTTP status code `301 Moved Permanently`. Redirection. This response code means the URI of requested resource has been changed. The new URI is usually included in the response.

ResponseCode **RESPONSE_FOUND** = `302`

HTTP status code `302 Found`. Temporary redirection. This response code means the URI of requested resource has been changed temporarily. New changes in the URI might be made in the future. Therefore, this same URI should be used by the client in future requests.

ResponseCode **RESPONSE_SEE_OTHER** = `303`

HTTP status code `303 See Other`. The server is redirecting the user agent to a different resource, as indicated by a URI in the Location header field, which is intended to provide an indirect response to the original request.

ResponseCode **RESPONSE_NOT_MODIFIED** = `304`

HTTP status code `304 Not Modified`. A conditional GET or HEAD request has been received and would have resulted in a 200 OK response if it were not for the fact that the condition evaluated to `false`.

ResponseCode **RESPONSE_USE_PROXY** = `305`

Deprecated: Many clients ignore this response code for security reasons. It is also deprecated by the HTTP standard.

HTTP status code `305 Use Proxy`.

ResponseCode **RESPONSE_SWITCH_PROXY** = `306`

Deprecated: Many clients ignore this response code for security reasons. It is also deprecated by the HTTP standard.

HTTP status code `306 Switch Proxy`.

ResponseCode **RESPONSE_TEMPORARY_REDIRECT** = `307`

HTTP status code `307 Temporary Redirect`. The target resource resides temporarily under a different URI and the user agent MUST NOT change the request method if it performs an automatic redirection to that URI.

ResponseCode **RESPONSE_PERMANENT_REDIRECT** = `308`

HTTP status code `308 Permanent Redirect`. The target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs.

ResponseCode **RESPONSE_BAD_REQUEST** = `400`

HTTP status code `400 Bad Request`. The request was invalid. The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, invalid request contents, or deceptive request routing).

ResponseCode **RESPONSE_UNAUTHORIZED** = `401`

HTTP status code `401 Unauthorized`. Credentials required. The request has not been applied because it lacks valid authentication credentials for the target resource.

ResponseCode **RESPONSE_PAYMENT_REQUIRED** = `402`

HTTP status code `402 Payment Required`. This response code is reserved for future use. Initial aim for creating this code was using it for digital payment systems, however this is not currently used.

ResponseCode **RESPONSE_FORBIDDEN** = `403`

HTTP status code `403 Forbidden`. The client does not have access rights to the content, i.e. they are unauthorized, so server is rejecting to give proper response. Unlike `401`, the client's identity is known to the server.

ResponseCode **RESPONSE_NOT_FOUND** = `404`

HTTP status code `404 Not Found`. The server can not find requested resource. Either the URL is not recognized or the endpoint is valid but the resource itself does not exist. May also be sent instead of 403 to hide existence of a resource if the client is not authorized.

ResponseCode **RESPONSE_METHOD_NOT_ALLOWED** = `405`

HTTP status code `405 Method Not Allowed`. The request's HTTP method is known by the server but has been disabled and cannot be used. For example, an API may forbid DELETE-ing a resource. The two mandatory methods, GET and HEAD, must never be disabled and should not return this error code.

ResponseCode **RESPONSE_NOT_ACCEPTABLE** = `406`

HTTP status code `406 Not Acceptable`. The target resource does not have a current representation that would be acceptable to the user agent, according to the proactive negotiation header fields received in the request. Used when negotiation content.

ResponseCode
RESPONSE_PROXY_AUTHENTICATION_REQUIRED = `407`

HTTP status code `407 Proxy Authentication Required`. Similar to 401 Unauthorized, but it indicates that the client needs to authenticate itself in order to use a proxy.

ResponseCode **RESPONSE_REQUEST_TIMEOUT** = `408`

HTTP status code `408 Request Timeout`. The server did not receive a complete request message within the time that it was prepared to wait.

ResponseCode **RESPONSE_CONFLICT** = `409`

HTTP status code `409 Conflict`. The request could not be completed due to a conflict with the current state of the target resource. This code is used in situations where the user might be able to resolve the conflict and resubmit the request.

ResponseCode **RESPONSE_GONE** = `410`

HTTP status code `410 Gone`. The target resource is no longer available at the origin server and this condition is likely permanent.

ResponseCode **RESPONSE_LENGTH_REQUIRED** = `411`

HTTP status code `411 Length Required`. The server refuses to accept the request without a defined Content-Length header.

ResponseCode **RESPONSE_PRECONDITION_FAILED** = `412`

HTTP status code `412 Precondition Failed`. One or more conditions given in the request header fields evaluated to `false` when tested on the server.

ResponseCode **RESPONSE_REQUEST_ENTITY_TOO_LARGE** = `413`

HTTP status code `413 Entity Too Large`. The server is refusing to process a request because the request payload is larger than the server is willing or able to process.

ResponseCode **RESPONSE_REQUEST_URI_TOO_LONG** = `414`

HTTP status code `414 Request-URI Too Long`. The server is refusing to service the request because the request-target is longer than the server is willing to interpret.

ResponseCode **RESPONSE_UNSUPPORTED_MEDIA_TYPE** = `415`

HTTP status code `415 Unsupported Media Type`. The origin server is refusing to service the request because the payload is in a format not supported by this method on the target resource.

ResponseCode **RESPONSE_REQUESTED_RANGE_NOT_SATISFIABLE** = `416`

HTTP status code `416 Requested Range Not Satisfiable`. None of the ranges in the request's Range header field overlap the current extent of the selected resource or the set of ranges requested has been rejected due to invalid ranges or an excessive request of small or overlapping ranges.

ResponseCode **RESPONSE_EXPECTATION_FAILED** = `417`

HTTP status code `417 Expectation Failed`. The expectation given in the request's Expect header field could not be met by at least one of the inbound servers.

ResponseCode **RESPONSE_IM_A_TEAPOT** = `418`

HTTP status code `418 I'm A Teapot`. Any attempt to brew coffee with a teapot should result in the error code "418 I'm a teapot". The resulting entity body MAY be short and stout.

ResponseCode **RESPONSE_MISDIRECTED_REQUEST** = `421`

HTTP status code `421 Misdirected Request`. The request was directed at a server that is not able to produce a response. This can be sent by a server that is not configured to produce responses for the combination of scheme and authority that are included in the request URI.

ResponseCode **RESPONSE_UNPROCESSABLE_ENTITY** = `422`

HTTP status code `422 Unprocessable Entity` (WebDAV). The server understands the content type of the request entity (hence a 415 Unsupported Media Type status code is inappropriate), and the syntax of the request entity is correct (thus a 400 Bad Request status code is inappropriate) but was unable to process the contained instructions.

ResponseCode **RESPONSE_LOCKED** = `423`

HTTP status code `423 Locked` (WebDAV). The source or destination resource of a method is locked.

ResponseCode **RESPONSE_FAILED_DEPENDENCY** = `424`

HTTP status code `424 Failed Dependency` (WebDAV). The method could not be performed on the resource because the requested action depended on another action and that action failed.

ResponseCode **RESPONSE_UPGRADE_REQUIRED** = `426`

HTTP status code `426 Upgrade Required`. The server refuses to perform the request using the current protocol but might be willing to do so after the client upgrades to a different protocol.

ResponseCode **RESPONSE_PRECONDITION_REQUIRED** = `428`

HTTP status code `428 Precondition Required`. The origin server requires the request to be conditional.

ResponseCode **RESPONSE_TOO_MANY_REQUESTS** = `429`

HTTP status code `429 Too Many Requests`. The user has sent too many requests in a given amount of time (see "rate limiting"). Back off and increase time between requests or try again later.

ResponseCode **RESPONSE_REQUEST_HEADER_FIELDS_TOO_LARGE** = `431`

HTTP status code `431 Request Header Fields Too Large`. The server is unwilling to process the request because its header fields are too large. The request MAY be resubmitted after reducing the size of the request header fields.

ResponseCode **RESPONSE_UNAVAILABLE_FOR_LEGAL_REASONS** = `451`

HTTP status code `451 Response Unavailable For Legal Reasons`. The server is denying access to the resource as a consequence of a legal demand.

ResponseCode **RESPONSE_INTERNAL_SERVER_ERROR** = `500`

HTTP status code `500 Internal Server Error`. The server encountered an unexpected condition that prevented it from fulfilling the request.

ResponseCode **RESPONSE_NOT_IMPLEMENTED** = `501`

HTTP status code `501 Not Implemented`. The server does not support the functionality required to fulfill the request.

ResponseCode **RESPONSE_BAD_GATEWAY** = `502`

HTTP status code `502 Bad Gateway`. The server, while acting as a gateway or proxy, received an invalid response from an inbound server it accessed while attempting to fulfill the request. Usually returned by load balancers or proxies.

ResponseCode **RESPONSE_SERVICE_UNAVAILABLE** = `503`

HTTP status code `503 Service Unavailable`. The server is currently unable to handle the request due to a temporary overload or scheduled maintenance, which will likely be alleviated after some delay. Try again later.

ResponseCode **RESPONSE_GATEWAY_TIMEOUT** = `504`

HTTP status code `504 Gateway Timeout`. The server, while acting as a gateway or proxy, did not receive a timely response from an upstream server it needed to access in order to complete the request. Usually returned by load balancers or proxies.

ResponseCode **RESPONSE_HTTP_VERSION_NOT_SUPPORTED** = `505`

HTTP status code `505 HTTP Version Not Supported`. The server does not support, or refuses to support, the major version of HTTP that was used in the request message.

ResponseCode **RESPONSE_VARIANT_ALSO_NEGOTIATES** = `506`

HTTP status code `506 Variant Also Negotiates`. The server has an internal configuration error: the chosen variant resource is configured to engage in transparent content negotiation itself, and is therefore not a proper end point in the negotiation process.

ResponseCode **RESPONSE_INSUFFICIENT_STORAGE** = `507`

HTTP status code `507 Insufficient Storage`. The method could not be performed on the resource because the server is unable to store the representation needed to successfully complete the request.

ResponseCode **RESPONSE_LOOP_DETECTED** = `508`

HTTP status code `508 Loop Detected`. The server terminated an operation because it encountered an infinite loop while processing a request with "Depth: infinity". This status indicates that the entire operation failed.

ResponseCode **RESPONSE_NOT_EXTENDED** = `510`

HTTP status code `510 Not Extended`. The policy for accessing the resource has not been met in the request. The server should send back all the information necessary for the client to issue an extended request.

ResponseCode **RESPONSE_NETWORK_AUTH_REQUIRED** = `511`

HTTP status code `511 Network Authentication Required`. The client needs to authenticate to gain network access.

Property Descriptions

bool **blocking_mode_enabled** = `false` 🔗

- void **set_blocking_mode**(value: bool)
- bool **is_blocking_mode_enabled**()

If `true`, execution will block until all data is read from the response.

StreamPeer **connection**

- void **set_connection**(value: StreamPeer)
- StreamPeer **get_connection**()

The connection to use for this client.

---

int **read_chunk_size** = `65536`

- void **set_read_chunk_size**(value: int)
- int **get_read_chunk_size**()

The size of the buffer used and maximum bytes to read per iteration. See read_response_body_chunk.

Method Descriptions

void **close**()

Closes the current connection, allowing reuse of this **HTTPClient**.

Error **connect_to_host**(host: String, port: int = -1, tls_options: TLSOptions = null)

Connects to a host. This needs to be done before any requests are sent.

If no `port` is specified (or `-1` is used), it is automatically set to 80 for HTTP and 443 for HTTPS. You can pass the optional `tls_options` parameter to customize the trusted certification authorities, or the common name verification when using HTTPS. See TLSOptions.client and TLSOptions.client_unsafe.

---

int **get_response_body_length**() const 🔗

Returns the response's body length.

Note: Some Web servers may not send a body length. In this case, the value returned will be `-1`. If using chunked transfer encoding, the body length will also be `-1`.

Note: This function always returns `-1` on the Web platform due to browsers limitations.

---

int **get_response_code**() const 🔗

Returns the response's HTTP status code.

PackedStringArray **get_response_headers**() 🔗

Returns the response headers.

Dictionary **get_response_headers_as_dictionary**() 🔗

Returns all response headers as a Dictionary of structure `{ "key": "value1; value2" }` where the case-sensitivity of the keys and values is kept like the server delivers it. A value is a simple String, this string can have more than one value where "; " is used as separator.

Example:

```
{
    "content-length": 12,
    "Content-Type": "application/json; charset=UTF-8",
}
```

Status **get_status**() const

Returns a Status constant. Need to call poll in order to get status updates.

bool **has_response**() const

If `true`, this **HTTPClient** has a response available.

bool **is_response_chunked**() const

If `true`, this **HTTPClient** has a response that is chunked.

Error **poll**()

This needs to be called in order to have any request processed. Check results with get_status.

String **query_string_from_dict**(fields: Dictionary)

Generates a GET/POST application/x-www-form-urlencoded style query string from a provided dictionary, e.g.:

GDScript

```
var fields = {"username": "user", "password": "pass"}
var query_string = http_client.query_string_from_dict(fields)
```

```
# Returns "username=user&password=pass"
```

C#

```
var fields = new Godot.Collections.Dictionary { { "username",
"user" }, { "password", "pass" } };
string queryString = httpClient.QueryStringFromDict(fields);
// Returns "username=user&password=pass"
```

Furthermore, if a key has a `null` value, only the key itself is added, without equal sign and value. If the value is an array, for each value in it a pair with the same key is added.

GDScript

```
var fields = {"single": 123, "not_valued": null, "multiple":
[22, 33, 44]}
var query_string = http_client.query_string_from_dict(fields)
# Returns
"single=123&not_valued&multiple=22&multiple=33&multiple=44"
```

C#

```
var fields = new Godot.Collections.Dictionary
{
    { "single", 123 },
    { "notValued", default },
    { "multiple", new Godot.Collections.Array { 22, 33, 44 } },
};
string queryString = httpClient.QueryStringFromDict(fields);
// Returns
"single=123&not_valued&multiple=22&multiple=33&multiple=44"
```

PackedByteArray **read_response_body_chunk**() 🔗

Reads one chunk from the response.

Error **request**(method: Method, url: String, headers: PackedStringArray, body: String = "") 🔗

Sends a request to the connected host.

The URL parameter is usually just the part after the host, so for `https://somehost.com/index.php`, it is `/index.php`. When sending requests to an HTTP proxy server, it should be an absolute URL. For METHOD_OPTIONS requests, `*` is also allowed. For METHOD_CONNECT requests, it should be the authority component (`host:port`).

Headers are HTTP request headers. For available HTTP methods, see Method.

To create a POST request with query strings to push to the server, do:

GDScript

```
var fields = {"username" : "user", "password" : "pass"}
var query_string = http_client.query_string_from_dict(fields)
var headers = ["Content-Type: application/x-www-form-urlencoded", "Content-Length: " + str(query_string.length())]
var result = http_client.request(http_client.METHOD_POST, "/index.php", headers, query_string)
```

C#

```
var fields = new Godot.Collections.Dictionary { { "username", "user" }, { "password", "pass" } };
string queryString = new HttpClient().QueryStringFromDict(fields);
string[] headers = { "Content-Type: application/x-www-form-urlencoded", $"Content-Length: {queryString.Length}" };
var result = new HttpClient().Request(HttpClient.Method.Post, "index.php", headers, queryString);
```

Note: The `body` parameter is ignored if `method` is METHOD_GET. This is because GET methods can't contain request data. As a workaround, you can pass request data as a query string in the URL. See String.uri_encode for an example.

Error **request_raw**(method: Method, url: String, headers: PackedStringArray, body: PackedByteArray) 🔗

Sends a raw request to the connected host.

The URL parameter is usually just the part after the host, so for `https://somehost.com/index.php`, it is `/index.php`. When sending requests to an HTTP proxy server, it should be an absolute URL. For METHOD_OPTIONS requests, `*` is also allowed. For METHOD_CONNECT requests, it should be the authority component (`host:port`).

Headers are HTTP request headers. For available HTTP methods, see Method.

Sends the body data raw, as a byte array and does not encode it in any way.

---

void **set_http_proxy**(host: String, port: int) 🔗

Sets the proxy server for HTTP requests.

The proxy server is unset if `host` is empty or `port` is -1.

---

void **set_https_proxy**(host: String, port: int) 🔗

Sets the proxy server for HTTPS requests.

The proxy server is unset if `host` is empty or `port` is -1.

ImageFormatLoader

Inherits: RefCounted **<** Object

Inherited By: ImageFormatLoaderExtension

Base class to add support for specific image formats.

Description

The engine supports multiple image formats out of the box (PNG, SVG, JPEG, WebP to name a few), but you can choose to implement support for additional image formats by extending ImageFormatLoaderExtension.

Enumerations

flags **LoaderFlags**: 🔗

LoaderFlags **FLAG_NONE** = `0`

There is currently no description for this enum. Please help us by contributing one!

LoaderFlags **FLAG_FORCE_LINEAR** = `1`

There is currently no description for this enum. Please help us by contributing one!

LoaderFlags **FLAG_CONVERT_COLORS** = `2`

There is currently no description for this enum. Please help us by contributing one!

ImageFormatLoaderExtension

Inherits: ImageFormatLoader **<** RefCounted **<** Object

Base class for creating ImageFormatLoader extensions (adding support for extra image formats).

Description

The engine supports multiple image formats out of the box (PNG, SVG, JPEG, WebP to name a few), but you can choose to implement support for additional image formats by extending this class.

Be sure to respect the documented return types and values. You should create an instance of it, and call add_format_loader to register that loader during the initialization phase.

Methods

PackedStringArray	_get_recognized_extensions() virtual const
Error	_load_image(image: Image, fileaccess: FileAccess, flags: BitField[LoaderFlags], scale: float) virtual
void	add_format_loader()

void	remove_format_loader()

Method Descriptions

PackedStringArray **_get_recognized_extensions**() virtual const

Returns the list of file extensions for this image format. Files with the given extensions will be treated as image file and loaded using this class.

Error **_load_image**(image: Image, fileaccess: FileAccess, flags: BitField[LoaderFlags], scale: float) virtual

Loads the content of `fileaccess` into the provided `image`.

void **add_format_loader**()

Add this format loader to the engine, allowing it to recognize the file extensions returned by _get_recognized_extensions.

void **remove_format_loader**()

Remove this format loader from the engine.

Input

Inherits: Object

A singleton for handling inputs.

Description

The **Input** singleton handles key presses, mouse buttons and movement, gamepads, and input actions. Actions and their events can be set in the **Input Map** tab in **Project > Project Settings**, or with the InputMap class.

Note: Input's methods reflect the global input state and are not affected by Control.accept_event or Viewport.set_input_as_handled, as those methods only deal with the way input is propagated in the SceneTree.

Tutorials

- Inputs documentation index
- 2D Dodge The Creeps Demo [https://godotengine.org/asset-library/asset/2712]
- 3D Voxel Demo [https://godotengine.org/asset-library/asset/2755]

Properties

bool	emulate_mouse_from_touch

bool	emulate_touch_from_mouse
MouseMode	mouse_mode
bool	use_accumulated_input

Methods

void	action_press(action: StringName, strength: float = 1.0)
void	action_release(action: StringName)
void	add_joy_mapping(mapping: String, update_existing: bool = false)
void	flush_buffered_events()
Vector3	get_accelerometer() const
float	get_action_raw_strength(action: StringName, exact_match: bool = false) const
float	get_action_strength(action: StringName, exact_match: bool =

	false) const
float	get_axis(negative_action: StringName, positive_action: StringName) const
Array[int]	get_connected_joypads()
CursorShape	get_current_cursor_shape() const
Vector3	get_gravity() const
Vector3	get_gyroscope() const
float	get_joy_axis(device: int, axis: JoyAxis) const
String	get_joy_guid(device: int) const
Dictionary	get_joy_info(device: int) const
String	get_joy_name(device: int)
float	get_joy_vibration_duration(device: int)
Vector2	get_joy_vibration_strength(device: int)

Vector2	get_last_mouse_screen_velocity()
Vector2	get_last_mouse_velocity()
Vector3	get_magnetometer() const
BitField[MouseButtonMask]	get_mouse_button_mask() const
Vector2	get_vector(negative_x: StringName, positive_x: StringName, negative_y: StringName, positive_y: StringName, deadzone: float = -1.0) const
bool	is_action_just_pressed(action: StringName, exact_match: bool = false) const
bool	is_action_just_released(action: StringName, exact_match: bool = false) const
bool	is_action_pressed(action: StringName, exact_match: bool = false) const
bool	is_anything_pressed() const

bool	is_joy_button_pressed(device: int, button: JoyButton) const
bool	is_joy_known(device: int)
bool	is_key_label_pressed(keycode: Key) const
bool	is_key_pressed(keycode: Key) const
bool	is_mouse_button_pressed(button: MouseButton) const
bool	is_physical_key_pressed(keycode: Key) const
void	parse_input_event(event: InputEvent)
void	remove_joy_mapping(guid: String)
void	set_accelerometer(value: Vector3)
void	set_custom_mouse_cursor(image: Resource, shape: CursorShape = 0, hotspot: Vector2 = Vector2(0, 0))

void	set_default_cursor_shape(shape: CursorShape = 0)
void	set_gravity(value: Vector3)
void	set_gyroscope(value: Vector3)
void	set_magnetometer(value: Vector3)
bool	should_ignore_device(vendor_id: int, product_id: int) const
void	start_joy_vibration(device: int, weak_magnitude: float, strong_magnitude: float, duration: float = 0)
void	stop_joy_vibration(device: int)
void	vibrate_handheld(duration_ms: int = 500, amplitude: float = -1.0)
void	warp_mouse(position: Vector2)

Signals

joy_connection_changed(device: int, connected: bool) 🔗

Emitted when a joypad device has been connected or disconnected.

Enumerations

enum **MouseMode**: 🔗

MouseMode **MOUSE_MODE_VISIBLE** = `0`

Makes the mouse cursor visible if it is hidden.

MouseMode **MOUSE_MODE_HIDDEN** = `1`

Makes the mouse cursor hidden if it is visible.

MouseMode **MOUSE_MODE_CAPTURED** = `2`

Captures the mouse. The mouse will be hidden and its position locked at the center of the window manager's window.

Note: If you want to process the mouse's movement in this mode, you need to use InputEventMouseMotion.relative.

MouseMode **MOUSE_MODE_CONFINED** = `3`

Confines the mouse cursor to the game window, and make it visible.

MouseMode **MOUSE_MODE_CONFINED_HIDDEN** = `4`

Confines the mouse cursor to the game window, and make it hidden.

---

enum **CursorShape**: 🔗

CursorShape **CURSOR_ARROW** = `0`

Arrow cursor. Standard, default pointing cursor.

CursorShape **CURSOR_IBEAM** = `1`

I-beam cursor. Usually used to show where the text cursor will appear when the mouse is clicked.

CursorShape **CURSOR_POINTING_HAND** = `2`

Pointing hand cursor. Usually used to indicate the pointer is over a link or other interactable item.

CursorShape **CURSOR_CROSS** = `3`

Cross cursor. Typically appears over regions in which a drawing operation can be performed or for selections.

CursorShape **CURSOR_WAIT** = `4`

Wait cursor. Indicates that the application is busy performing an operation, and that it cannot be used during the operation (e.g. something is blocking its main thread).

CursorShape **CURSOR_BUSY** = `5`

Busy cursor. Indicates that the application is busy performing an operation, and that it is still usable during the operation.

CursorShape **CURSOR_DRAG** = `6`

Drag cursor. Usually displayed when dragging something.

Note: Windows lacks a dragging cursor, so CURSOR_DRAG is the same as CURSOR_MOVE for this platform.

CursorShape **CURSOR_CAN_DROP** = `7`

Can drop cursor. Usually displayed when dragging something to indicate that it can be dropped at the current position.

CursorShape **CURSOR_FORBIDDEN** = `8`

Forbidden cursor. Indicates that the current action is forbidden (for example, when dragging something) or that the control at a position is disabled.

CursorShape **CURSOR_VSIZE** = `9`

Vertical resize mouse cursor. A double-headed vertical arrow. It tells the user they can resize the window or the panel vertically.

CursorShape **CURSOR_HSIZE** = `10`

Horizontal resize mouse cursor. A double-headed horizontal arrow. It tells the user they can resize the window or the panel horizontally.

CursorShape **CURSOR_BDIAGSIZE** = `11`

Window resize mouse cursor. The cursor is a double-headed arrow that goes from the bottom left to the top right. It tells the user they can resize the window or the panel both horizontally and vertically.

CursorShape **CURSOR_FDIAGSIZE** = `12`

Window resize mouse cursor. The cursor is a double-headed arrow that goes from the top left to the bottom right, the opposite of CURSOR_BDIAGSIZE. It tells the user they can resize the window or the panel both horizontally and vertically.

CursorShape **CURSOR_MOVE** = `13`

Move cursor. Indicates that something can be moved.

CursorShape **CURSOR_VSPLIT** = `14`

Vertical split mouse cursor. On Windows, it's the same as CURSOR_VSIZE.

CursorShape **CURSOR_HSPLIT** = `15`

Horizontal split mouse cursor. On Windows, it's the same as CURSOR_HSIZE.

CursorShape **CURSOR_HELP** = `16`

Help cursor. Usually a question mark.

Property Descriptions

bool **emulate_mouse_from_touch**

- void **set_emulate_mouse_from_touch**(value: bool)
- bool **is_emulating_mouse_from_touch**()

If `true`, sends mouse input events when tapping or swiping on the touchscreen. See also ProjectSettings.input_devices/pointing/emulate_mouse_from_touch.

---

bool **emulate_touch_from_mouse**

- void **set_emulate_touch_from_mouse**(value: bool)
- bool **is_emulating_touch_from_mouse**()

If `true`, sends touch input events when clicking or dragging the mouse. See also ProjectSettings.input_devices/pointing/emulate_touch_from_mouse.

MouseMode **mouse_mode**

- void **set_mouse_mode**(value: MouseMode)
- MouseMode **get_mouse_mode**()

Controls the mouse mode. See MouseMode for more information.

---

bool **use_accumulated_input** 🔗

- void **set_use_accumulated_input**(value: bool)
- bool **is_using_accumulated_input**()

If `true`, similar input events sent by the operating system are accumulated. When input accumulation is enabled, all input events generated during a frame will be merged and emitted when the frame is done rendering. Therefore, this limits the number of input method calls per second to the rendering FPS.

Input accumulation can be disabled to get slightly more precise/reactive input at the cost of increased CPU usage. In applications where drawing freehand lines is required, input accumulation should generally be disabled while the user is drawing the line to get results that closely follow the actual input.

Note: Input accumulation is *enabled* by default.

Method Descriptions

void **action_press**(action: StringName, strength: float = 1.0) 🔗

This will simulate pressing the specified action.

The strength can be used for non-boolean actions, it's ranged between 0 and 1 representing the intensity of the given action.

Note: This method will not cause any Node._input calls. It is intended to be used with is_action_pressed and is_action_just_pressed. If you want to simulate `_input`, use parse_input_event instead.

void **action_release**(action: StringName)

If the specified action is already pressed, this will release it.

void **add_joy_mapping**(mapping: String, update_existing: bool = false)

Adds a new mapping entry (in SDL2 format) to the mapping database. Optionally update already connected devices.

void **flush_buffered_events**()

Sends all input events which are in the current buffer to the game loop. These events may have been buffered as a result of accumulated input (use_accumulated_input) or agile input flushing (ProjectSettings.input_devices/buffering/agile_event_flushing).

The engine will already do this itself at key execution points (at least once per frame). However, this can be useful in advanced cases where you want precise control over the timing of event handling.

Vector3 **get_accelerometer**() const

Returns the acceleration in m/s² of the device's accelerometer sensor, if the device has one. Otherwise, the method returns Vector3.ZERO.

Note this method returns an empty Vector3 when running from the editor even when your device has an accelerometer. You must export your project to a supported device to read values from the accelerometer.

Note: This method only works on Android and iOS. On other platforms, it always returns Vector3.ZERO.

---

float **get_action_raw_strength**(action: StringName, exact_match: bool = false) const

Returns a value between 0 and 1 representing the raw intensity of the given action, ignoring the action's deadzone. In most cases, you should use get_action_strength instead.

If `exact_match` is `false`, it ignores additional input modifiers for InputEventKey and InputEventMouseButton events, and the direction for InputEventJoypadMotion events.

---

float **get_action_strength**(action: StringName, exact_match: bool = false) const

Returns a value between 0 and 1 representing the intensity of the given action. In a joypad, for example, the further away the axis (analog sticks or L2, R2 triggers) is from the dead zone, the closer the value will be to 1. If the action is mapped to a control that has no axis such as the keyboard, the value returned will be 0 or 1.

If `exact_match` is `false`, it ignores additional input modifiers for InputEventKey and InputEventMouseButton events, and the direction for InputEventJoypadMotion events.

---

float **get_axis**(negative_action: StringName, positive_action: StringName) const

Get axis input by specifying two actions, one negative and one positive.

This is a shorthand for writing `Input.get_action_strength("positive_action") - Input.get_action_strength("negative_action")`.

Array[int] **get_connected_joypads**()

Returns an Array containing the device IDs of all currently connected joypads.

CursorShape **get_current_cursor_shape**() const

Returns the currently assigned cursor shape (see CursorShape).

Vector3 **get_gravity**() const

Returns the gravity in m/s² of the device's accelerometer sensor, if the device has one. Otherwise, the method returns Vector3.ZERO.

Note: This method only works on Android and iOS. On other platforms, it always returns Vector3.ZERO.

Vector3 **get_gyroscope**() const

Returns the rotation rate in rad/s around a device's X, Y, and Z axes of the gyroscope sensor, if the device has one. Otherwise, the method returns Vector3.ZERO.

Note: This method only works on Android and iOS. On other platforms, it always returns Vector3.ZERO.

---

float **get_joy_axis**(device: int, axis: JoyAxis) const 🔗

Returns the current value of the joypad axis at given index (see JoyAxis).

String **get_joy_guid**(device: int) const 🔗

Returns an SDL2-compatible device GUID on platforms that use gamepad remapping, e.g. `030000004c050000c405000000010000`. Returns `"Default Gamepad"` otherwise. Godot uses the SDL2 game controller database [https://github.com/gabomdq/SDL_GameControllerDB] to determine gamepad names and mappings based on this GUID.

Dictionary **get_joy_info**(device: int) const 🔗

Returns a dictionary with extra platform-specific information about the device, e.g. the raw gamepad name from the OS or the Steam Input index.

On Windows the dictionary contains the following fields:

`xinput_index`: The index of the controller in the XInput system.

On Linux:

`raw_name`: The name of the controller as it came from the OS, before getting renamed by the godot controller database.

`vendor_id`: The USB vendor ID of the device.

`product_id`: The USB product ID of the device.

`steam_input_index`: The Steam Input gamepad index, if the device is not a Steam Input device this key won't be present.

String **get_joy_name**(device: int)

Returns the name of the joypad at the specified device index, e.g. `PS4 Controller`. Godot uses the SDL2 game controller database [https://github.com/gabomdq/SDL_GameControllerDB] to determine gamepad names.

---

float **get_joy_vibration_duration**(device: int)

Returns the duration of the current vibration effect in seconds.

Vector2 **get_joy_vibration_strength**(device: int)

Returns the strength of the joypad vibration: x is the strength of the weak motor, and y is the strength of the strong motor.

Vector2 **get_last_mouse_screen_velocity**()

Returns the last mouse velocity in screen coordinates. To provide a precise and jitter-free velocity, mouse velocity is only calculated every 0.1s. Therefore, mouse velocity will lag mouse movements.

Vector2 **get_last_mouse_velocity**()

Returns the last mouse velocity. To provide a precise and jitter-free velocity, mouse velocity is only calculated every 0.1s. Therefore, mouse velocity will lag mouse movements.

Vector3 **get_magnetometer**() const 🔗

Returns the magnetic field strength in micro-Tesla for all axes of the device's magnetometer sensor, if the device has one. Otherwise, the method returns Vector3.ZERO.

Note: This method only works on Android and iOS. On other platforms, it always returns Vector3.ZERO.

BitField[MouseButtonMask] **get_mouse_button_mask**() const 🔗

Returns mouse buttons as a bitmask. If multiple mouse buttons are pressed at the same time, the bits are added together. Equivalent to DisplayServer.mouse_get_button_state.

Vector2 **get_vector**(negative_x: StringName, positive_x: StringName, negative_y: StringName, positive_y: StringName, deadzone: float = -1.0) const 🔗

Gets an input vector by specifying four actions for the positive and negative X and Y axes.

This method is useful when getting vector input, such as from a joystick, directional pad, arrows, or WASD. The vector has its length limited to 1 and has a circular deadzone, which is useful for using vector input as movement.

By default, the deadzone is automatically calculated from the average of the action deadzones. However, you can override the deadzone to be whatever you want (on the range of 0 to 1).

---

bool **is_action_just_pressed**(action: StringName, exact_match: bool = false) const 🔗

Returns `true` when the user has *started* pressing the action event in the current frame or physics tick. It will only return `true` on the frame or tick that the user pressed down the button.

This is useful for code that needs to run only once when an action is pressed, instead of every frame while it's pressed.

If `exact_match` is `false`, it ignores additional input modifiers for InputEventKey and InputEventMouseButton events, and the direction for InputEventJoypadMotion events.

Note: Returning `true` does not imply that the action is *still* pressed. An action can be pressed and released again rapidly, and `true` will still be returned so as not to miss input.

Note: Due to keyboard ghosting, is_action_just_pressed may return `false` even if one of the action's keys is pressed. See Input examples in the documentation for more information.

Note: During input handling (e.g. Node._input), use InputEvent.is_action_pressed instead to query the action state of the current event.

---

bool **is_action_just_released**(action: StringName, exact_match: bool = false) const 🔗

Returns `true` when the user *stops* pressing the action event in the current frame or physics tick. It will only return `true` on the frame or tick that the user releases the button.

Note: Returning `true` does not imply that the action is *still* not pressed. An action can be released and pressed again rapidly, and `true` will still be returned so as not to miss input.

If `exact_match` is `false`, it ignores additional input modifiers for InputEventKey and InputEventMouseButton events, and the direction for InputEventJoypadMotion events.

Note: During input handling (e.g. Node._input), use InputEvent.is_action_released instead to query the action state of the current event.

---

bool **is_action_pressed**(action: StringName, exact_match: bool = false) const 🔗

Returns `true` if you are pressing the action event.

If `exact_match` is `false`, it ignores additional input modifiers for InputEventKey and InputEventMouseButton events, and the direction for InputEventJoypadMotion events.

Note: Due to keyboard ghosting, is_action_pressed may return `false` even if one of the action's keys is pressed. See Input examples in the documentation for more information.

---

bool **is_anything_pressed**() const 🔗

Returns `true` if any action, key, joypad button, or mouse button is being pressed. This will also return `true` if any action is simulated via code by calling action_press.

---

bool **is_joy_button_pressed**(device: int, button: JoyButton) const 🔗

Returns `true` if you are pressing the joypad button (see JoyButton).

---

bool **is_joy_known**(device: int)

Returns `true` if the system knows the specified device. This means that it sets all button and axis indices. Unknown joypads are not expected to match these constants, but you can still retrieve events from them.

---

bool **is_key_label_pressed**(keycode: Key) const

Returns `true` if you are pressing the key with the `keycode` printed on it. You can pass a Key constant or any Unicode character code.

---

bool **is_key_pressed**(keycode: Key) const

Returns `true` if you are pressing the Latin key in the current keyboard layout. You can pass a Key constant.

is_key_pressed is only recommended over is_physical_key_pressed in non-game applications. This ensures that shortcut keys behave as expected depending on the user's keyboard layout, as keyboard shortcuts are generally dependent on the keyboard layout in non-game applications. If in doubt, use is_physical_key_pressed.

Note: Due to keyboard ghosting, is_key_pressed may return `false` even if one of the action's keys is pressed. See Input examples in the documentation for more information.

---

bool **is_mouse_button_pressed**(button: MouseButton) const

Returns `true` if you are pressing the mouse button specified with MouseButton.

---

bool **is_physical_key_pressed**(keycode: Key) const 🔗

Returns `true` if you are pressing the key in the physical location on the 101/102-key US QWERTY keyboard. You can pass a Key constant.

is_physical_key_pressed is recommended over is_key_pressed for in-game actions, as it will make `W`/`A`/`S`/`D` layouts work regardless of the user's keyboard layout. is_physical_key_pressed will also ensure that the top row number keys work on any keyboard layout. If in doubt, use is_physical_key_pressed.

Note: Due to keyboard ghosting, is_physical_key_pressed may return `false` even if one of the action's keys is pressed. See Input examples in the documentation for more information.

---

void **parse_input_event**(event: InputEvent) 🔗

Feeds an InputEvent to the game. Can be used to artificially trigger input events from code. Also generates Node._input calls.

Example:

GDScript

```
var cancel_event = InputEventAction.new()
cancel_event.action = "ui_cancel"
cancel_event.pressed = true
Input.parse_input_event(cancel_event)
```

C#

```
var cancelEvent = new InputEventAction();
cancelEvent.Action = "ui_cancel";
cancelEvent.Pressed = true;
Input.ParseInputEvent(cancelEvent);
```

Note: Calling this function has no influence on the operating system. So for example sending an InputEventMouseMotion will not move the OS mouse cursor to the specified position (use warp_mouse instead) and sending `Alt/Cmd + Tab` as InputEventKey won't toggle between active windows.

void **remove_joy_mapping**(guid: String)

Removes all mappings from the internal database that match the given GUID.

void **set_accelerometer**(value: Vector3)

Sets the acceleration value of the accelerometer sensor. Can be used for debugging on devices without a hardware sensor, for example in an editor on a PC.

Note: This value can be immediately overwritten by the hardware sensor value on Android and iOS.

void **set_custom_mouse_cursor**(image: Resource, shape: CursorShape = 0, hotspot: Vector2 = Vector2(0, 0))

Sets a custom mouse cursor image, which is only visible inside the game window. The hotspot can also be specified. Passing `null` to the image parameter resets to the system cursor. See CursorShape for the list of shapes.

`image` can be either Texture2D or Image and its size must be lower than or equal to 256×256. To avoid rendering issues, sizes lower than or equal to 128×128 are recommended.

`hotspot` must be within `image`'s size.

Note: AnimatedTextures aren't supported as custom mouse cursors. If using an AnimatedTexture, only the first frame will be displayed.

Note: The **Lossless**, **Lossy** or **Uncompressed** compression modes are recommended. The **Video RAM** compression mode can be used, but it will be decompressed on the CPU, which means loading times are slowed down and no memory is saved compared to lossless modes.

Note: On the web platform, the maximum allowed cursor image size is 128×128. Cursor images larger than 32×32 will also only be displayed if the mouse cursor image is entirely located within the page for security reasons [https://chromestatus.com/feature/5825971391299584].

void **set_default_cursor_shape**(shape: CursorShape = 0) 🔗

Sets the default cursor shape to be used in the viewport instead of CURSOR_ARROW.

Note: If you want to change the default cursor shape for Control's nodes, use Control.mouse_default_cursor_shape instead.

Note: This method generates an InputEventMouseMotion to update cursor immediately.

void **set_gravity**(value: Vector3) 🔗

Sets the gravity value of the accelerometer sensor. Can be used for debugging on devices without a hardware sensor, for example in an editor on a PC.

Note: This value can be immediately overwritten by the hardware sensor value on Android and iOS.

void **set_gyroscope**(value: Vector3) 🔗

Sets the value of the rotation rate of the gyroscope sensor. Can be used for debugging on devices without a hardware sensor, for example in an editor on a PC.

Note: This value can be immediately overwritten by the hardware sensor value on Android and iOS.

---

void **set_magnetometer**(value: Vector3) 🔗

Sets the value of the magnetic field of the magnetometer sensor. Can be used for debugging on devices without a hardware sensor, for example in an editor on a PC.

Note: This value can be immediately overwritten by the hardware sensor value on Android and iOS.

---

bool **should_ignore_device**(vendor_id: int, product_id: int) const 🔗

Queries whether an input device should be ignored or not. Devices can be ignored by setting the environment variable `SDL_GAMECONTROLLER_IGNORE_DEVICES`. Read the SDL documentation [https://wiki.libsdl.org/SDL2] for more information.

Note: Some 3rd party tools can contribute to the list of ignored devices. For example, *SteamInput* creates virtual devices from physical devices for remapping purposes. To avoid handling the same input device twice, the original device is added to the ignore list.

---

void **start_joy_vibration**(device: int, weak_magnitude: float, strong_magnitude: float, duration: float = 0)

Starts to vibrate the joypad. Joypads usually come with two rumble motors, a strong and a weak one. `weak_magnitude` is the strength of the weak motor (between 0 and 1) and `strong_magnitude` is the strength of the strong motor (between 0 and 1). `duration` is the duration of the effect in seconds (a duration of 0 will try to play the vibration indefinitely). The vibration can be stopped early by calling stop_joy_vibration.

Note: Not every hardware is compatible with long effect durations; it is recommended to restart an effect if it has to be played for more than a few seconds.

Note: For macOS, vibration is only supported in macOS 11 and later.

---

void **stop_joy_vibration**(device: int)

Stops the vibration of the joypad started with start_joy_vibration.

---

void **vibrate_handheld**(duration_ms: int = 500, amplitude: float = -1.0)

Vibrate the handheld device for the specified duration in milliseconds.

`amplitude` is the strength of the vibration, as a value between `0.0` and `1.0`. If set to `-1.0`, the default vibration strength of the device is used.

Note: This method is implemented on Android, iOS, and Web. It has no effect on other platforms.

Note: For Android, vibrate_handheld requires enabling the `VIBRATE` permission in the export preset. Otherwise, vibrate_handheld will have no effect.

Note: For iOS, specifying the duration is only supported in iOS 13 and later.

Note: For Web, the amplitude cannot be changed.

Note: Some web browsers such as Safari and Firefox for Android do not support vibrate_handheld.

---

void **warp_mouse**(position: Vector2) 🔗

Sets the mouse position to the specified vector, provided in pixels and relative to an origin at the upper left corner of the currently focused Window Manager game window.

Mouse position is clipped to the limits of the screen resolution, or to the limits of the game window if MouseMode is set to MOUSE_MODE_CONFINED or MOUSE_MODE_CONFINED_HIDDEN.

Note: warp_mouse is only supported on Windows, macOS and Linux. It has no effect on Android, iOS and Web.

InputMap

Inherits: Object

A singleton that manages all InputEventActions.

Description

Manages all InputEventAction which can be created/modified from the project settings menu **Project > Project Settings > Input Map** or in code with add_action and action_add_event. See Node._input.

Tutorials

- Using InputEvent: InputMap

Methods

void	action_add_event(action: StringName, event: InputEvent)
void	action_erase_event(action: StringName, event: InputEvent)
void	action_erase_events(action: StringName)

float	action_get_deadzone(action: StringName)
Array[InputEvent]	action_get_events(action: StringName)
bool	action_has_event(action: StringName, event: InputEvent)
void	action_set_deadzone(action: StringName, deadzone: float)
void	add_action(action: StringName, deadzone: float = 0.5)
void	erase_action(action: StringName)
bool	event_is_action(event: InputEvent, action: StringName, exact_match: bool = false) const
Array[StringName]	get_actions()
bool	has_action(action: StringName) const
void	load_from_project_settings()

Method Descriptions

void **action_add_event**(action: StringName, event: InputEvent) 🔗

Adds an InputEvent to an action. This InputEvent will trigger the action.

void **action_erase_event**(action: StringName, event: InputEvent) 🔗

Removes an InputEvent from an action.

void **action_erase_events**(action: StringName) 🔗

Removes all events from an action.

float **action_get_deadzone**(action: StringName) 🔗

Returns a deadzone value for the action.

Array[InputEvent] **action_get_events**(action: StringName) 🔗

Returns an array of InputEvents associated with a given action.

Note: When used in the editor (e.g. a tool script or EditorPlugin), this method will return events for the editor action. If you want to access your project's input binds from the editor, read the `input/*` settings from ProjectSettings.

bool **action_has_event**(action: StringName, event: InputEvent) 🔗

Returns `true` if the action has the given InputEvent associated with it.

---

void **action_set_deadzone**(action: StringName, deadzone: float) 🔗

Sets a deadzone value for the action.

---

void **add_action**(action: StringName, deadzone: float = 0.5) 🔗

Adds an empty action to the **InputMap** with a configurable `deadzone`.

An InputEvent can then be added to this action with action_add_event.

---

void **erase_action**(action: StringName) 🔗

Removes an action from the **InputMap**.

---

bool **event_is_action**(event: InputEvent, action: StringName, exact_match: bool = false) const 🔗

Returns `true` if the given event is part of an existing action. This method ignores keyboard modifiers if the given InputEvent is not pressed (for proper release detection). See action_has_event if you don't want this behavior.

If `exact_match` is `false`, it ignores additional input modifiers for InputEventKey and InputEventMouseButton events, and the direction for InputEventJoypadMotion events.

Array[StringName] **get_actions**() 🔗

Returns an array of all actions in the **InputMap**.

bool **has_action**(action: StringName) const 🔗

Returns `true` if the **InputMap** has a registered action with the given name.

void **load_from_project_settings**() 🔗

Clears all InputEventAction in the **InputMap** and load it anew from ProjectSettings.

IntervalTweener

Inherits: Tweener **<** RefCounted **<** Object

Creates an idle interval in a Tween animation.

Description

IntervalTweener is used to make delays in a tweening sequence. See Tween.tween_interval for more usage information.

Note: Tween.tween_interval is the only correct way to create **IntervalTweener**. Any **IntervalTweener** created manually will not function correctly.

IP

Inherits: Object

Internet protocol (IP) support functions such as DNS resolution.

Description

IP contains support functions for the Internet Protocol (IP). TCP/IP support is in different classes (see StreamPeerTCP and TCPServer). IP provides DNS hostname resolution support, both blocking and threaded.

Methods

void	clear_cache(hostname: String = "")
void	erase_resolve_item(id: int)
PackedStringArray	get_local_addresses() const
Array[Dictionary]	get_local_interfaces() const
String	get_resolve_item_address(id: int) const
Array	get_resolve_item_addresses(id: int) const

ResolverStatus	get_resolve_item_status(id: int) const
String	resolve_hostname(host: String, ip_type: Type = 3)
PackedStringArray	resolve_hostname_addresses(host: String, ip_type: Type = 3)
int	resolve_hostname_queue_item(host: String, ip_type: Type = 3)

Enumerations

enum **ResolverStatus**: 🔗

ResolverStatus **RESOLVER_STATUS_NONE** = `0`

DNS hostname resolver status: No status.

ResolverStatus **RESOLVER_STATUS_WAITING** = `1`

DNS hostname resolver status: Waiting.

ResolverStatus **RESOLVER_STATUS_DONE** = `2`

DNS hostname resolver status: Done.

ResolverStatus **RESOLVER_STATUS_ERROR** = `3`

DNS hostname resolver status: Error.

enum **Type**: 🔗

Type **TYPE_NONE** = `0`

Address type: None.

Type **TYPE_IPV4** = `1`

Address type: Internet protocol version 4 (IPv4).

Type **TYPE_IPV6** = `2`

Address type: Internet protocol version 6 (IPv6).

Type **TYPE_ANY** = `3`

Address type: Any.

Constants

RESOLVER_MAX_QUERIES = `256` 🔗

Maximum number of concurrent DNS resolver queries allowed, RESOLVER_INVALID_ID is returned if exceeded.

RESOLVER_INVALID_ID = `-1` 🔗

Invalid ID constant. Returned if RESOLVER_MAX_QUERIES is exceeded.

Method Descriptions

void **clear_cache**(hostname: String = "") 🔗

Removes all of a `hostname`'s cached references. If no `hostname` is given, all cached IP addresses are removed.

---

void **erase_resolve_item**(id: int) 🔗

Removes a given item `id` from the queue. This should be used to free a queue after it has completed to enable more queries to happen.

PackedStringArray **get_local_addresses**() const 🔗

Returns all the user's current IPv4 and IPv6 addresses as an array.

Array[Dictionary] **get_local_interfaces**() const 🔗

Returns all network adapters as an array.

Each adapter is a dictionary of the form:

```
{
    "index": "1", # Interface index.
    "name": "eth0", # Interface name.
    "friendly": "Ethernet One", # A friendly name (might be
empty).
    "addresses": ["192.168.1.101"], # An array of IP addresses
associated to this interface.
}
```

String **get_resolve_item_address**(id: int) const 🔗

Returns a queued hostname's IP address, given its queue `id`. Returns an empty string on error or if resolution hasn't happened yet (see get_resolve_item_status).

Array **get_resolve_item_addresses**(id: int) const

Returns resolved addresses, or an empty array if an error happened or resolution didn't happen yet (see get_resolve_item_status).

ResolverStatus **get_resolve_item_status**(id: int) const

Returns a queued hostname's status as a ResolverStatus constant, given its queue `id`.

String **resolve_hostname**(host: String, ip_type: Type = 3)

Returns a given hostname's IPv4 or IPv6 address when resolved (blocking-type method). The address type returned depends on the Type constant given as `ip_type`.

PackedStringArray **resolve_hostname_addresses**(host: String, ip_type: Type = 3)

Resolves a given hostname in a blocking way. Addresses are returned as an Array of IPv4 or IPv6 addresses depending on `ip_type`.

---

int **resolve_hostname_queue_item**(host: String, ip_type: Type = 3)

Creates a queue item to resolve a hostname to an IPv4 or IPv6 address depending on the Type constant given as `ip_type`. Returns the queue ID if successful, or RESOLVER_INVALID_ID on error.

JavaClass

Inherits: RefCounted **<** Object

Represents an object from the Java Native Interface.

Description

Represents an object from the Java Native Interface. It is returned from JavaClassWrapper.wrap.

Note: This class only works on Android. For any other build, this class does nothing.

Note: This class is not to be confused with JavaScriptObject.

JavaClassWrapper

Inherits: Object

Provides access to the Java Native Interface.

Description

The JavaClassWrapper singleton provides a way for the Godot application to send and receive data through the Java Native Interface [https://developer.android.com/training/articles/perf-jni] (JNI).

Note: This singleton is only available in Android builds.

Methods

JavaClass	wrap(name: String)

Method Descriptions

JavaClass **wrap**(name: String) 🔗

Wraps a class defined in Java, and returns it as a JavaClass Object type that Godot can interact with.

Note: This method only works on Android. On every other platform, this method does nothing and returns an empty JavaClass.

JavaScriptBridge

Inherits: Object

Singleton that connects the engine with the browser's JavaScript context in Web export.

Description

The JavaScriptBridge singleton is implemented only in the Web export. It's used to access the browser's JavaScript context. This allows interaction with embedding pages or calling third-party JavaScript APIs.

Note: This singleton can be disabled at build-time to improve security. By default, the JavaScriptBridge singleton is enabled. Official export templates also have the JavaScriptBridge singleton enabled. See Compiling for the Web in the documentation for more information.

Tutorials

- Exporting for the Web: Calling JavaScript from script

Methods

JavaScriptObject	create_callback(callable: Callable)
Variant	create_object(object: String, ...) vararg

void	download_buffer(buffer: PackedByteArray, name: String, mime: String = "application/octet-stream")
Variant	eval(code: String, use_global_execution_context: bool = false)
void	force_fs_sync()
JavaScriptObject	get_interface(interface: String)
bool	pwa_needs_update() const
Error	pwa_update()

Signals

pwa_update_available() 🔗

Emitted when an update for this progressive web app has been detected but is waiting to be activated because a previous version is active. See pwa_update to force the update to take place immediately.

Method Descriptions

JavaScriptObject **create_callback**(callable: Callable)

Creates a reference to a Callable that can be used as a callback by JavaScript. The reference must be kept until the callback happens, or it won't be called at all. See JavaScriptObject for usage.

Variant **create_object**(object: String, ...) vararg

Creates a new JavaScript object using the `new` constructor. The `object` must a valid property of the JavaScript `window`. See JavaScriptObject for usage.

---

void **download_buffer**(buffer: PackedByteArray, name: String, mime: String = "application/octet-stream")

Prompts the user to download a file containing the specified `buffer`. The file will have the given `name` and `mime` type.

Note: The browser may override the MIME type [https://en.wikipedia.org/wiki/Media_type] provided based on the file `name`'s extension.

Note: Browsers might block the download if download_buffer is not being called from a user interaction (e.g. button click).

Note: Browsers might ask the user for permission or block the download if multiple download requests are made in a quick succession.

Variant **eval**(code: String, use_global_execution_context: bool = false)

Execute the string `code` as JavaScript code within the browser window. This is a call to the actual global JavaScript function `eval()`.

If `use_global_execution_context` is `true`, the code will be evaluated in the global execution context. Otherwise, it is evaluated in the execution context of a function within the engine's runtime environment.

---

void **force_fs_sync**() 🔗

Force synchronization of the persistent file system (when enabled).

Note: This is only useful for modules or extensions that can't use FileAccess to write files.

JavaScriptObject **get_interface**(interface: String) 🔗

Returns an interface to a JavaScript object that can be used by scripts. The `interface` must be a valid property of the JavaScript `window`. The callback must accept a single Array argument, which will contain the JavaScript `arguments`. See JavaScriptObject for usage.

---

bool **pwa_needs_update**() const 🔗

Returns `true` if a new version of the progressive web app is waiting to be activated.

Note: Only relevant when exported as a Progressive Web App.

Error **pwa_update**() 🔗

Performs the live update of the progressive web app. Forcing the new version to be installed and the page to be reloaded.

Note: Your application will be **reloaded in all browser tabs**.

Note: Only relevant when exported as a Progressive Web App and pwa_needs_update returns `true`.

JavaScriptObject

Inherits: RefCounted **<** Object

A wrapper class for web native JavaScript objects.

Description

JavaScriptObject is used to interact with JavaScript objects retrieved or created via JavaScriptBridge.get_interface, JavaScriptBridge.create_object, or JavaScriptBridge.create_callback.

Example:

```
extends Node

var _my_js_callback =
JavaScriptBridge.create_callback(myCallback) # This reference
must be kept
var console = JavaScriptBridge.get_interface("console")

func _init():
    var buf = JavaScriptBridge.create_object("ArrayBuffer", 10)
# new ArrayBuffer(10)
    print(buf) # prints [JavaScriptObject:OBJECT_ID]
    var uint8arr = JavaScriptBridge.create_object("Uint8Array",
buf) # new Uint8Array(buf)
    uint8arr[1] = 255
    prints(uint8arr[1], uint8arr.byteLength) # prints 255 10
    console.log(uint8arr) # prints in browser console
"Uint8Array(10) [ 0, 255, 0, 0, 0, 0, 0, 0, 0, 0 ]"

    # Equivalent of JavaScriptBridge:
Array.from(uint8arr).forEach(myCallback)

JavaScriptBridge.get_interface("Array").from(uint8arr).forEach(
_my_js_callback)
```

```
func myCallback(args):
    # Will be called with the parameters passed to the
"forEach" callback
    # [0, 0, [JavaScriptObject:1173]]
    # [255, 1, [JavaScriptObject:1173]]
    # ...
    # [0, 9, [JavaScriptObject:1180]]
    print(args)
```

Note: Only available in the Web platform.

JNISingleton

Inherits: Object

Singleton that connects the engine with Android plugins to interface with native Android code.

Description

The JNISingleton is implemented only in the Android export. It's used to call methods and connect signals from an Android plugin written in Java or Kotlin. Methods and signals can be called and connected to the JNISingleton as if it is a Node. See Java Native Interface - Wikipedia [https://en.wikipedia.org/wiki/Java_Native_Interface] for more information.

Tutorials

- Creating Android plugins

JSONRPC

Inherits: Object

A helper to handle dictionaries which look like JSONRPC documents.

Description

JSON-RPC [https://www.jsonrpc.org/] is a standard which wraps a method call in a JSON object. The object has a particular structure and identifies which method is called, the parameters to that function, and carries an ID to keep track of responses. This class implements that standard on top of Dictionary; you will have to convert between a Dictionary and JSON with other functions.

Methods

Dictionary	make_notification(method: String, params: Variant)
Dictionary	make_request(method: String, params: Variant, id: Variant)
Dictionary	make_response(result: Variant, id: Variant)
Dictionary	make_response_error(code: int, message: String, id: Variant = null) const

Variant	process_action(action: Variant, recurse: bool = false)
String	process_string(action: String)
void	set_scope(scope: String, target: Object)

Enumerations

enum **ErrorCode**:

ErrorCode **PARSE_ERROR** = `-32700`

The request could not be parsed as it was not valid by JSON standard (JSON.parse failed).

ErrorCode **INVALID_REQUEST** = `-32600`

A method call was requested but the request's format is not valid.

ErrorCode **METHOD_NOT_FOUND** = `-32601`

A method call was requested but no function of that name existed in the JSONRPC subclass.

ErrorCode **INVALID_PARAMS** = `-32602`

A method call was requested but the given method parameters are not valid. Not used by the built-in JSONRPC.

ErrorCode **INTERNAL_ERROR** = `-32603`

An internal error occurred while processing the request. Not used by the built-in JSONRPC.

Method Descriptions

Dictionary **make_notification**(method: String, params: Variant)

Returns a dictionary in the form of a JSON-RPC notification. Notifications are one-shot messages which do not expect a response.

- `method`: Name of the method being called.
- `params`: An array or dictionary of parameters being passed to the method.

Dictionary **make_request**(method: String, params: Variant, id: Variant)

Returns a dictionary in the form of a JSON-RPC request. Requests are sent to a server with the expectation of a response. The ID field is used for the server to specify which exact request it is responding to.

- `method`: Name of the method being called.
- `params`: An array or dictionary of parameters being passed to the method.
- `id`: Uniquely identifies this request. The server is expected to send a response with the same ID.

Dictionary **make_response**(result: Variant, id: Variant)

When a server has received and processed a request, it is expected to send a response. If you did not want a response then you need to have sent a Notification instead.

- `result`: The return value of the function which was called.
- `id`: The ID of the request this response is targeted to.

Dictionary **make_response_error**(code: int, message: String, id: Variant = null) const 🔗

Creates a response which indicates a previous reply has failed in some way.

- `code`: The error code corresponding to what kind of error this is. See the ErrorCode constants.
- `message`: A custom message about this error.
- `id`: The request this error is a response to.

Variant **process_action**(action: Variant, recurse: bool = false) 🔗

Given a Dictionary which takes the form of a JSON-RPC request: unpack the request and run it. Methods are resolved by looking at the field called "method" and looking for an equivalently named function in the JSONRPC object. If one is found that method is called.

To add new supported methods extend the JSONRPC class and call process_action on your subclass.

`action`: The action to be run, as a Dictionary in the form of a JSON-RPC request or notification.

String **process_string**(action: String) 🔗

There is currently no description for this method. Please help us by contributing one!

void **set_scope**(scope: String, target: Object) 🔗

There is currently no description for this method. Please help us by contributing one!

KinematicCollision2D

Inherits: RefCounted **<** Object

Holds collision data from the movement of a PhysicsBody2D.

Description

Holds collision data from the movement of a PhysicsBody2D, usually from PhysicsBody2D.move_and_collide. When a PhysicsBody2D is moved, it stops if it detects a collision with another body. If a collision is detected, a **KinematicCollision2D** object is returned.

The collision data includes the colliding object, the remaining motion, and the collision position. This data can be used to determine a custom response to the collision.

Methods

float	get_angle(up_direction: Vector2 = Vector2(0, -1)) const
Object	get_collider() const
int	get_collider_id() const
RID	get_collider_rid() const
Object	get_collider_shape() const

int	get_collider_shape_index() const
Vector2	get_collider_velocity() const
float	get_depth() const
Object	get_local_shape() const
Vector2	get_normal() const
Vector2	get_position() const
Vector2	get_remainder() const
Vector2	get_travel() const

Method Descriptions

float **get_angle**(up_direction: Vector2 = Vector2(0, -1)) const 🔗

Returns the collision angle according to `up_direction`, which is Vector2.UP by default. This value is always positive.

Object **get_collider**() const 🔗

Returns the colliding body's attached Object.

---

int **get_collider_id**() const

Returns the unique instance ID of the colliding body's attached Object. See Object.get_instance_id.

RID **get_collider_rid**() const

Returns the colliding body's RID used by the PhysicsServer2D.

Object **get_collider_shape**() const

Returns the colliding body's shape.

---

int **get_collider_shape_index**() const

Returns the colliding body's shape index. See CollisionObject2D.

Vector2 **get_collider_velocity**() const

Returns the colliding body's velocity.

---

float **get_depth**() const

Returns the colliding body's length of overlap along the collision normal.

Object **get_local_shape**() const

Returns the moving object's colliding shape.

Vector2 **get_normal**() const 🔗

Returns the colliding body's shape's normal at the point of collision.

Vector2 **get_position**() const 🔗

Returns the point of collision in global coordinates.

Vector2 **get_remainder**() const 🔗

Returns the moving object's remaining movement vector.

Vector2 **get_travel**() const 🔗

Returns the moving object's travel before collision.

KinematicCollision3D

Inherits: RefCounted **<** Object

Holds collision data from the movement of a PhysicsBody3D.

Description

Holds collision data from the movement of a PhysicsBody3D, usually from PhysicsBody3D.move_and_collide. When a PhysicsBody3D is moved, it stops if it detects a collision with another body. If a collision is detected, a **KinematicCollision3D** object is returned.

The collision data includes the colliding object, the remaining motion, and the collision position. This data can be used to determine a custom response to the collision.

Methods

float	get_angle(collision_index: int = 0, up_direction: Vector3 = Vector3(0, 1, 0)) const
Object	get_collider(collision_index: int = 0) const
int	get_collider_id(collision_index: int = 0) const
RID	get_collider_rid(collision_index: int = 0) const

Object	get_collider_shape(collision_index: int = 0) const
int	get_collider_shape_index(collision_index: int = 0) const
Vector3	get_collider_velocity(collision_index: int = 0) const
int	get_collision_count() const
float	get_depth() const
Object	get_local_shape(collision_index: int = 0) const
Vector3	get_normal(collision_index: int = 0) const
Vector3	get_position(collision_index: int = 0) const
Vector3	get_remainder() const
Vector3	get_travel() const

Method Descriptions

float **get_angle**(collision_index: int = 0, up_direction: Vector3 = Vector3(0, 1, 0)) const 🔗

Returns the collision angle according to `up_direction`, which is Vector3.UP by default. This value is always positive.

Object **get_collider**(collision_index: int = 0) const

Returns the colliding body's attached Object given a collision index (the deepest collision by default).

---

int **get_collider_id**(collision_index: int = 0) const

Returns the unique instance ID of the colliding body's attached Object given a collision index (the deepest collision by default). See Object.get_instance_id.

RID **get_collider_rid**(collision_index: int = 0) const

Returns the colliding body's RID used by the PhysicsServer3D given a collision index (the deepest collision by default).

Object **get_collider_shape**(collision_index: int = 0) const

Returns the colliding body's shape given a collision index (the deepest collision by default).

---

int **get_collider_shape_index**(collision_index: int = 0) const

Returns the colliding body's shape index given a collision index (the deepest collision by default). See CollisionObject3D.

Vector3 **get_collider_velocity**(collision_index: int = 0) const

Returns the colliding body's velocity given a collision index (the deepest collision by default).

---

int **get_collision_count**() const 🔗

Returns the number of detected collisions.

---

float **get_depth**() const 🔗

Returns the colliding body's length of overlap along the collision normal.

Object **get_local_shape**(collision_index: int = 0) const 🔗

Returns the moving object's colliding shape given a collision index (the deepest collision by default).

Vector3 **get_normal**(collision_index: int = 0) const 🔗

Returns the colliding body's shape's normal at the point of collision given a collision index (the deepest collision by default).

Vector3 **get_position**(collision_index: int = 0) const 🔗

Returns the point of collision in global coordinates given a collision index (the deepest collision by default).

Vector3 **get_remainder**() const 🔗

Returns the moving object's remaining movement vector.

Vector3 **get_travel**() const

Returns the moving object's travel before collision.

Lightmapper

Inherits: RefCounted **<** Object

Inherited By: LightmapperRD

Abstract class extended by lightmappers, for use in LightmapGI.

Description

This class should be extended by custom lightmapper classes. Lightmappers can then be used with LightmapGI to provide fast baked global illumination in 3D.

Godot contains a built-in GPU-based lightmapper LightmapperRD that uses compute shaders, but custom lightmappers can be implemented by C++ modules.

LightmapperRD

Inherits: Lightmapper **<** RefCounted **<** Object

The built-in GPU-based lightmapper for use with LightmapGI.

Description

LightmapperRD ("RD" stands for RenderingDevice) is the built-in GPU-based lightmapper for use with LightmapGI. On most dedicated GPUs, it can bake lightmaps much faster than most CPU-based lightmappers. LightmapperRD uses compute shaders to bake lightmaps, so it does not require CUDA or OpenCL libraries to be installed to be usable.

Note: Only usable when using the Vulkan backend (Forward+ or Mobile), not OpenGL.

MainLoop

Inherits: Object

Inherited By: SceneTree

Abstract base class for the game's main loop.

Description

MainLoop is the abstract base class for a Godot project's game loop. It is inherited by SceneTree, which is the default game loop implementation used in Godot projects, though it is also possible to write and use one's own **MainLoop** subclass instead of the scene tree.

Upon the application start, a **MainLoop** implementation must be provided to the OS; otherwise, the application will exit. This happens automatically (and a SceneTree is created) unless a **MainLoop** Script is provided from the command line (with e.g. `godot -s my_loop.gd`) or the "Main Loop Type" project setting is overwritten.

Here is an example script implementing a simple **MainLoop**:

GDScript

```
class_name CustomMainLoop
extends MainLoop

var time_elapsed = 0

func _initialize():
    print("Initialized:")
    print("  Starting time: %s" % str(time_elapsed))
```

```
func _process(delta):
    time_elapsed += delta
    # Return true to end the main loop.
    return Input.get_mouse_button_mask() != 0 ||
Input.is_key_pressed(KEY_ESCAPE)

func _finalize():
    print("Finalized:")
    print("  End time: %s" % str(time_elapsed))
```

C#

```
using Godot;

[GlobalClass]
public partial class CustomMainLoop : MainLoop
{
    private double _timeElapsed = 0;

    public override void _Initialize()
    {
        GD.Print("Initialized:");
        GD.Print($"  Starting Time: {_timeElapsed}");
    }

    public override bool _Process(double delta)
    {
        _timeElapsed += delta;
        // Return true to end the main loop.
        return Input.GetMouseButtonMask() != 0 ||
Input.IsKeyPressed(Key.Escape);
    }

    private void _Finalize()
    {
        GD.Print("Finalized:");
        GD.Print($"  End Time: {_timeElapsed}");
    }
}
```

Methods

void	_finalize() virtual
void	_initialize() virtual
bool	_physics_process(delta: float) virtual
bool	_process(delta: float) virtual

Signals

on_request_permissions_result(permission: String, granted: bool)

Emitted when a user responds to a permission request.

Constants

NOTIFICATION_OS_MEMORY_WARNING = `2009`

Notification received from the OS when the application is exceeding its allocated memory.

Specific to the iOS platform.

NOTIFICATION_TRANSLATION_CHANGED = `2010`

Notification received when translations may have changed. Can be triggered by the user changing the locale. Can be used to respond to language changes, for example to change the UI strings on the fly. Useful when working with the built-in translation support, like Object.tr.

NOTIFICATION_WM_ABOUT = `2011` 🔗

Notification received from the OS when a request for "About" information is sent.

Specific to the macOS platform.

NOTIFICATION_CRASH = `2012` 🔗

Notification received from Godot's crash handler when the engine is about to crash.

Implemented on desktop platforms if the crash handler is enabled.

NOTIFICATION_OS_IME_UPDATE = `2013` 🔗

Notification received from the OS when an update of the Input Method Engine occurs (e.g. change of IME cursor position or composition string).

Specific to the macOS platform.

NOTIFICATION_APPLICATION_RESUMED = `2014` 🔗

Notification received from the OS when the application is resumed.

Specific to the Android and iOS platforms.

NOTIFICATION_APPLICATION_PAUSED = `2015` 🔗

Notification received from the OS when the application is paused.

Specific to the Android and iOS platforms.

Note: On iOS, you only have approximately 5 seconds to finish a task started by this signal. If you go over this allotment, iOS will kill the app instead of pausing it.

NOTIFICATION_APPLICATION_FOCUS_IN = `2016`

Notification received from the OS when the application is focused, i.e. when changing the focus from the OS desktop or a thirdparty application to any open window of the Godot instance.

Implemented on desktop and mobile platforms.

NOTIFICATION_APPLICATION_FOCUS_OUT = `2017`

Notification received from the OS when the application is defocused, i.e. when changing the focus from any open window of the Godot instance to the OS desktop or a thirdparty application.

Implemented on desktop and mobile platforms.

NOTIFICATION_TEXT_SERVER_CHANGED = `2018`

Notification received when text server is changed.

Method Descriptions

void **_finalize**() virtual

Called before the program exits.

---

void **_initialize**() virtual

Called once during initialization.

---

bool **_physics_process**(delta: float) virtual

Called each physics frame with the time since the last physics frame as argument (`delta`, in seconds). Equivalent to Node._physics_process.

If implemented, the method must return a boolean value. `true` ends the main loop, while `false` lets it proceed to the next frame.

---

bool **_process**(delta: float) virtual

Called each process (idle) frame with the time since the last process frame as argument (in seconds). Equivalent to Node._process.

If implemented, the method must return a boolean value. `true` ends the main loop, while `false` lets it proceed to the next frame.

Marshalls

Inherits: Object

Data transformation (marshaling) and encoding helpers.

Description

Provides data transformation and encoding utility functions.

Methods

PackedByteArray	base64_to_raw(base64_str: String)
String	base64_to_utf8(base64_str: String)
Variant	base64_to_variant(base64_str: String, allow_objects: bool = false)
String	raw_to_base64(array: PackedByteArray)
String	utf8_to_base64(utf8_str: String)
String	variant_to_base64(variant: Variant, full_objects: bool = false)

Method Descriptions

PackedByteArray **base64_to_raw**(base64_str: String) 🔗

Returns a decoded PackedByteArray corresponding to the Base64-encoded string `base64_str`.

String **base64_to_utf8**(base64_str: String) 🔗

Returns a decoded string corresponding to the Base64-encoded string `base64_str`.

Variant **base64_to_variant**(base64_str: String, allow_objects: bool = false) 🔗

Returns a decoded Variant corresponding to the Base64-encoded string `base64_str`. If `allow_objects` is `true`, decoding objects is allowed.

Internally, this uses the same decoding mechanism as the @GlobalScope.bytes_to_var method.

Warning: Deserialized objects can contain code which gets executed. Do not use this option if the serialized object comes from untrusted sources to avoid potential security threats such as remote code execution.

String **raw_to_base64**(array: PackedByteArray) 🔗

Returns a Base64-encoded string of a given PackedByteArray.

String **utf8_to_base64**(utf8_str: String) 🔗

Returns a Base64-encoded string of the UTF-8 string `utf8_str`.

String **variant_to_base64**(variant: Variant, full_objects: bool = false) 🔗

Returns a Base64-encoded string of the Variant `variant`. If `full_objects` is `true`, encoding objects is allowed (and can potentially include code).

Internally, this uses the same encoding mechanism as the @GlobalScope.var_to_bytes method.

MeshConvexDecompositionSettings

Inherits: RefCounted **<** Object

Parameters to be used with a Mesh convex decomposition operation.

Description

Parameters to be used with a Mesh convex decomposition operation.

Properties

bool	convex_hull_approximation	`true`
int	convex_hull_downsampling	`4`
float	max_concavity	`1.0`
int	max_convex_hulls	`1`
int	max_num_vertices_per_convex_hull	`32`
float	min_volume_per_convex_hull	`0.0001`

Mode	mode	`0`
bool	normalize_mesh	`false`
int	plane_downsampling	`4`
bool	project_hull_vertices	`true`
int	resolution	`10000`
float	revolution_axes_clipping_bias	`0.05`
float	symmetry_planes_clipping_bias	`0.05`

Enumerations

enum **Mode**: 🔗

Mode **CONVEX_DECOMPOSITION_MODE_VOXEL** = `0`

Constant for voxel-based approximate convex decomposition.

Mode **CONVEX_DECOMPOSITION_MODE_TETRAHEDRON** = `1`

Constant for tetrahedron-based approximate convex decomposition.

Property Descriptions

bool **convex_hull_approximation** = `true`

- void **set_convex_hull_approximation**(value: bool)
- bool **get_convex_hull_approximation**()

If `true`, uses approximation for computing convex hulls.

---

int **convex_hull_downsampling** = `4`

- void **set_convex_hull_downsampling**(value: int)
- int **get_convex_hull_downsampling**()

Controls the precision of the convex-hull generation process during the clipping plane selection stage. Ranges from `1` to `16`.

---

float **max_concavity** = `1.0`

- void **set_max_concavity**(value: float)
- float **get_max_concavity**()

Maximum concavity. Ranges from `0.0` to `1.0`.

---

int **max_convex_hulls** = `1`

- void **set_max_convex_hulls**(value: int)
- int **get_max_convex_hulls**()

The maximum number of convex hulls to produce from the merge operation.

---

int **max_num_vertices_per_convex_hull** = `32`

- void **set_max_num_vertices_per_convex_hull**(value: int)
- int **get_max_num_vertices_per_convex_hull**()

Controls the maximum number of triangles per convex-hull. Ranges from `4` to `1024`.

---

float **min_volume_per_convex_hull** = `0.0001`

- void **set_min_volume_per_convex_hull**(value: float)
- float **get_min_volume_per_convex_hull**()

Controls the adaptive sampling of the generated convex-hulls. Ranges from `0.0` to `0.01`.

Mode **mode** = `0`

- void **set_mode**(value: Mode)
- Mode **get_mode**()

Mode for the approximate convex decomposition.

---

bool **normalize_mesh** = `false`

- void **set_normalize_mesh**(value: bool)
- bool **get_normalize_mesh**()

If `true`, normalizes the mesh before applying the convex decomposition.

---

int **plane_downsampling** = `4`

- void **set_plane_downsampling**(value: int)
- int **get_plane_downsampling**()

Controls the granularity of the search for the "best" clipping plane. Ranges from `1` to `16`.

---

int **project_hull_vertices** = `true`

- void **set_project_hull_vertices**(value: bool)
- bool **get_project_hull_vertices**()

If `true`, projects output convex hull vertices onto the original source mesh to increase floating-point accuracy of the results.

---

int **resolution** = `10000`

- void **set_resolution**(value: int)
- int **get_resolution**()

Maximum number of voxels generated during the voxelization stage.

---

float **revolution_axes_clipping_bias** = `0.05`

- void **set_revolution_axes_clipping_bias**(value: float)
- float **get_revolution_axes_clipping_bias**()

Controls the bias toward clipping along revolution axes. Ranges from `0.0` to `1.0`.

---

float **symmetry_planes_clipping_bias** = `0.05`

- void **set_symmetry_planes_clipping_bias**(value: float)
- float **get_symmetry_planes_clipping_bias**()

Controls the bias toward clipping along symmetry planes. Ranges from `0.0` to `1.0`.

MeshDataTool

Inherits: RefCounted **<** Object

Helper tool to access and edit Mesh data.

Description

MeshDataTool provides access to individual vertices in a Mesh. It allows users to read and edit vertex data of meshes. It also creates an array of faces and edges.

To use MeshDataTool, load a mesh with create_from_surface. When you are finished editing the data commit the data to a mesh with commit_to_surface.

Below is an example of how MeshDataTool may be used.

GDScript

```
var mesh = ArrayMesh.new()
mesh.add_surface_from_arrays(Mesh.PRIMITIVE_TRIANGLES,
BoxMesh.new().get_mesh_arrays())
var mdt = MeshDataTool.new()
mdt.create_from_surface(mesh, 0)
for i in range(mdt.get_vertex_count()):
    var vertex = mdt.get_vertex(i)
    # In this example we extend the mesh by one unit, which
results in separated faces as it is flat shaded.
    vertex += mdt.get_vertex_normal(i)
    # Save your change.
    mdt.set_vertex(i, vertex)
mesh.clear_surfaces()
mdt.commit_to_surface(mesh)
var mi = MeshInstance.new()
```

```
mi.mesh = mesh
add_child(mi)
```

C#

```
var mesh = new ArrayMesh();
mesh.AddSurfaceFromArrays(Mesh.PrimitiveType.Triangles, new
BoxMesh().GetMeshArrays());
var mdt = new MeshDataTool();
mdt.CreateFromSurface(mesh, 0);
for (var i = 0; i < mdt.GetVertexCount(); i++)
{
    Vector3 vertex = mdt.GetVertex(i);
    // In this example we extend the mesh by one unit, which
results in separated faces as it is flat shaded.
    vertex += mdt.GetVertexNormal(i);
    // Save your change.
    mdt.SetVertex(i, vertex);
}
mesh.ClearSurfaces();
mdt.CommitToSurface(mesh);
var mi = new MeshInstance();
mi.Mesh = mesh;
AddChild(mi);
```

See also ArrayMesh, ImmediateMesh and SurfaceTool for procedural geometry generation.

Note: Godot uses clockwise winding order [https://learnopengl.com/Advanced-OpenGL/Face-culling] for front faces of triangle primitive modes.

Tutorials

- Using the MeshDataTool

Methods

void	clear()
Error	commit_to_surface(mesh: ArrayMesh, compression_flags: int = 0)
Error	create_from_surface(mesh: ArrayMesh, surface: int)
int	get_edge_count() const
PackedInt32Array	get_edge_faces(idx: int) const
Variant	get_edge_meta(idx: int) const
int	get_edge_vertex(idx: int, vertex: int) const
int	get_face_count() const
int	get_face_edge(idx: int, edge: int) const
Variant	get_face_meta(idx: int) const
Vector3	get_face_normal(idx: int) const
int	get_face_vertex(idx: int, vertex: int) const

int	get_format() const
Material	get_material() const
Vector3	get_vertex(idx: int) const
PackedInt32Array	get_vertex_bones(idx: int) const
Color	get_vertex_color(idx: int) const
int	get_vertex_count() const
PackedInt32Array	get_vertex_edges(idx: int) const
PackedInt32Array	get_vertex_faces(idx: int) const
Variant	get_vertex_meta(idx: int) const
Vector3	get_vertex_normal(idx: int) const
Plane	get_vertex_tangent(idx: int) const
Vector2	get_vertex_uv(idx: int) const
Vector2	get_vertex_uv2(idx: int) const

PackedFloat32Array	get_vertex_weights(idx: int) const
void	set_edge_meta(idx: int, meta: Variant)
void	set_face_meta(idx: int, meta: Variant)
void	set_material(material: Material)
void	set_vertex(idx: int, vertex: Vector3)
void	set_vertex_bones(idx: int, bones: PackedInt32Array)
void	set_vertex_color(idx: int, color: Color)
void	set_vertex_meta(idx: int, meta: Variant)
void	set_vertex_normal(idx: int, normal: Vector3)
void	set_vertex_tangent(idx: int, tangent: Plane)
void	set_vertex_uv(idx: int, uv: Vector2)
void	set_vertex_uv2(idx: int, uv2: Vector2)

void	set_vertex_weights(idx: int, weights: PackedFloat32Array)

Method Descriptions

void **clear**() 🔗

Clears all data currently in MeshDataTool.

Error **commit_to_surface**(mesh: ArrayMesh, compression_flags: int = 0) 🔗

Adds a new surface to specified Mesh with edited data.

Error **create_from_surface**(mesh: ArrayMesh, surface: int) 🔗

Uses specified surface of given Mesh to populate data for MeshDataTool.

Requires Mesh with primitive type Mesh.PRIMITIVE_TRIANGLES.

---

int **get_edge_count**() const 🔗

Returns the number of edges in this Mesh.

PackedInt32Array **get_edge_faces**(idx: int) const 🔗

Returns array of faces that touch given edge.

Variant **get_edge_meta**(idx: int) const

Returns meta information assigned to given edge.

int **get_edge_vertex**(idx: int, vertex: int) const

Returns index of specified vertex connected to given edge.

Vertex argument can only be 0 or 1 because edges are comprised of two vertices.

int **get_face_count**() const

Returns the number of faces in this Mesh.

int **get_face_edge**(idx: int, edge: int) const

Returns specified edge associated with given face.

Edge argument must be either 0, 1, or 2 because a face only has three edges.

Variant **get_face_meta**(idx: int) const

Returns the metadata associated with the given face.

Vector3 **get_face_normal**(idx: int) const

Calculates and returns the face normal of the given face.

int **get_face_vertex**(idx: int, vertex: int) const

Returns the specified vertex index of the given face.

Vertex argument must be either 0, 1, or 2 because faces contain three vertices.

Example:

GDScript

```
var index = mesh_data_tool.get_face_vertex(0, 1) # Gets the index of the second vertex of the first face.
var position = mesh_data_tool.get_vertex(index)
var normal = mesh_data_tool.get_vertex_normal(index)
```

C#

```
int index = meshDataTool.GetFaceVertex(0, 1); // Gets the index of the second vertex of the first face.
Vector3 position = meshDataTool.GetVertex(index);
Vector3 normal = meshDataTool.GetVertexNormal(index);
```

---

int **get_format**() const

Returns the Mesh's format as a combination of the ArrayFormat flags. For example, a mesh containing both vertices and normals would return a format of `3` because Mesh.ARRAY_FORMAT_VERTEX is `1` and Mesh.ARRAY_FORMAT_NORMAL is `2`.

Material **get_material**() const

Returns the material assigned to the Mesh.

Vector3 **get_vertex**(idx: int) const

Returns the position of the given vertex.

PackedInt32Array **get_vertex_bones**(idx: int) const

Returns the bones of the given vertex.

Color **get_vertex_color**(idx: int) const

Returns the color of the given vertex.

---

int **get_vertex_count**() const

Returns the total number of vertices in Mesh.

PackedInt32Array **get_vertex_edges**(idx: int) const

Returns an array of edges that share the given vertex.

PackedInt32Array **get_vertex_faces**(idx: int) const

Returns an array of faces that share the given vertex.

Variant **get_vertex_meta**(idx: int) const

Returns the metadata associated with the given vertex.

Vector3 **get_vertex_normal**(idx: int) const

Returns the normal of the given vertex.

Plane **get_vertex_tangent**(idx: int) const

Returns the tangent of the given vertex.

Vector2 **get_vertex_uv**(idx: int) const

Returns the UV of the given vertex.

Vector2 **get_vertex_uv2**(idx: int) const

Returns the UV2 of the given vertex.

PackedFloat32Array **get_vertex_weights**(idx: int) const

Returns bone weights of the given vertex.

---

void **set_edge_meta**(idx: int, meta: Variant)

Sets the metadata of the given edge.

---

void **set_face_meta**(idx: int, meta: Variant)

Sets the metadata of the given face.

---

void **set_material**(material: Material)

Sets the material to be used by newly-constructed Mesh.

---

void **set_vertex**(idx: int, vertex: Vector3)

Sets the position of the given vertex.

---

void **set_vertex_bones**(idx: int, bones: PackedInt32Array) 🔗

Sets the bones of the given vertex.

void **set_vertex_color**(idx: int, color: Color) 🔗

Sets the color of the given vertex.

void **set_vertex_meta**(idx: int, meta: Variant) 🔗

Sets the metadata associated with the given vertex.

void **set_vertex_normal**(idx: int, normal: Vector3) 🔗

Sets the normal of the given vertex.

void **set_vertex_tangent**(idx: int, tangent: Plane) 🔗

Sets the tangent of the given vertex.

void **set_vertex_uv**(idx: int, uv: Vector2) 🔗

Sets the UV of the given vertex.

void **set_vertex_uv2**(idx: int, uv2: Vector2) 🔗

Sets the UV2 of the given vertex.

void **set_vertex_weights**(idx: int, weights: PackedFloat32Array) 🔗

Sets the bone weights of the given vertex.

MethodTweener

Inherits: Tweener **<** RefCounted **<** Object

Interpolates an abstract value and supplies it to a method called over time.

Description

MethodTweener is similar to a combination of CallbackTweener and PropertyTweener. It calls a method providing an interpolated value as a parameter. See Tween.tween_method for more usage information.

The tweener will finish automatically if the callback's target object is freed.

Note: Tween.tween_method is the only correct way to create **MethodTweener**. Any **MethodTweener** created manually will not function correctly.

Methods

MethodTweener	set_delay(delay: float)
MethodTweener	set_ease(ease: EaseType)
MethodTweener	set_trans(trans: TransitionType)

Method Descriptions

MethodTweener **set_delay**(delay: float) 🔗

Sets the time in seconds after which the **MethodTweener** will start interpolating. By default there's no delay.

MethodTweener **set_ease**(ease: EaseType) 🔗

Sets the type of used easing from EaseType. If not set, the default easing is used from the Tween that contains this Tweener.

MethodTweener **set_trans**(trans: TransitionType) 🔗

Sets the type of used transition from TransitionType. If not set, the default transition is used from the Tween that contains this Tweener.

MobileVRInterface

Inherits: XRInterface **<** RefCounted **<** Object

Generic mobile VR implementation.

Description

This is a generic mobile VR implementation where you need to provide details about the phone and HMD used. It does not rely on any existing framework. This is the most basic interface we have. For the best effect, you need a mobile phone with a gyroscope and accelerometer.

Note that even though there is no positional tracking, the camera will assume the headset is at a height of 1.85 meters. You can change this by setting eye_height.

You can initialize this interface as follows:

```
var interface = XRServer.find_interface("Native mobile")
if interface and interface.initialize():
    get_viewport().use_xr = true
```

Properties

float	display_to_lens	`4.0`
float	display_width	`14.5`

float	eye_height	`1.85`
float	iod	`6.0`
float	k1	`0.215`
float	k2	`0.215`
Rect2	offset_rect	`Rect2(0, 0, 1, 1)`
float	oversample	`1.5`
float	vrs_min_radius	`20.0`
float	vrs_strength	`1.0`
PlayAreaMode	xr_play_area_mode	`1` (overrides XRInterface)

Property Descriptions

float **display_to_lens** = `4.0` 🔗

- void **set_display_to_lens**(value: float)
- float **get_display_to_lens**()

The distance between the display and the lenses inside of the device in centimeters.

---

float **display_width** = `14.5`

- void **set_display_width**(value: float)
- float **get_display_width**()

The width of the display in centimeters.

---

float **eye_height** = `1.85`

- void **set_eye_height**(value: float)
- float **get_eye_height**()

The height at which the camera is placed in relation to the ground (i.e. XROrigin3D node).

---

float **iod** = `6.0`

- void **set_iod**(value: float)
- float **get_iod**()

The interocular distance, also known as the interpupillary distance. The distance between the pupils of the left and right eye.

---

float **k1** = `0.215`

- void **set_k1**(value: float)
- float **get_k1**()

The k1 lens factor is one of the two constants that define the strength of the lens used and directly influences the lens distortion effect.

---

float **k2** = `0.215` 🔗

- void **set_k2**(value: float)
- float **get_k2**()

The k2 lens factor, see k1.

Rect2 **offset_rect** = `Rect2(0, 0, 1, 1)` 🔗

- void **set_offset_rect**(value: Rect2)
- Rect2 **get_offset_rect**()

Set the offset rect relative to the area being rendered. A length of 1 represents the whole rendering area on that axis.

---

float **oversample** = `1.5` 🔗

- void **set_oversample**(value: float)
- float **get_oversample**()

The oversample setting. Because of the lens distortion we have to render our buffers at a higher resolution then the screen can natively handle. A value between 1.5 and 2.0 often provides good results but at the cost of performance.

---

float **vrs_min_radius** = `20.0` 🔗

- void **set_vrs_min_radius**(value: float)
- float **get_vrs_min_radius**()

The minimum radius around the focal point where full quality is guaranteed if VRS is used as a percentage of screen size.

Note: Mobile and Forward+ renderers only. Requires Viewport.vrs_mode to be set to Viewport.VRS_XR.

---

float **vrs_strength** = `1.0` 🔗

- void **set_vrs_strength**(value: float)
- float **get_vrs_strength**()

The strength used to calculate the VRS density map. The greater this value, the more noticeable VRS is. This improves performance at the cost of quality.

Note: Mobile and Forward+ renderers only. Requires Viewport.vrs_mode to be set to Viewport.VRS_XR.

MovieWriter

Inherits: Object

Abstract class for non-real-time video recording encoders.

Description

Godot can record videos with non-real-time simulation. Like the `--fixed-fps` command line argument, this forces the reported `delta` in Node._process functions to be identical across frames, regardless of how long it actually took to render the frame. This can be used to record high-quality videos with perfect frame pacing regardless of your hardware's capabilities.

Godot has 2 built-in **MovieWriter**s:

- AVI container with MJPEG for video and uncompressed audio (`.avi` file extension). Lossy compression, medium file sizes, fast encoding. The lossy compression quality can be adjusted by changing ProjectSettings.editor/movie_writer/mjpeg_quality. The resulting file can be viewed in most video players, but it must be converted to another format for viewing on the web or by Godot with VideoStreamPlayer. MJPEG does not support transparency. AVI output is currently limited to a file of 4 GB in size at most.
- PNG image sequence for video and WAV for audio (`.png` file extension). Lossless compression, large file sizes, slow encoding. Designed to be encoded to a video file with another tool such as FFmpeg [https://ffmpeg.org/] after recording. Transparency is currently not supported, even if the root viewport is set to be transparent.

If you need to encode to a different format or pipe a stream through third-party software, you can extend the **MovieWriter** class to create your own movie writers. This should typically be done using GDExtension for performance reasons.

Editor usage: A default movie file path can be specified in ProjectSettings.editor/movie_writer/movie_file. Alternatively, for running single scenes, a `movie_file` metadata can be added to the root node, specifying the path to a movie file that will be used when recording that scene. Once a path is set, click the video reel icon in the top-right corner of the editor to enable Movie Maker mode, then run any scene as usual. The engine will start recording as soon as the splash screen is finished, and it will only stop recording when the engine quits. Click the video reel icon again to disable Movie Maker mode. Note that toggling Movie Maker mode does not affect project instances that are already running.

Note: MovieWriter is available for use in both the editor and exported projects, but it is *not* designed for use by end users to record videos while playing. Players wishing to record gameplay videos should install tools such as OBS Studio [https://obsproject.com/] or SimpleScreenRecorder [https://www.maartenbaert.be/simplescreenrecorder/] instead.

Methods

int	_get_audio_mix_rate() virtual const
SpeakerMode	_get_audio_speaker_mode() virtual const
bool	_handles_file(path: String) virtual const

Error	_write_begin(movie_size: Vector2i, fps: int, base_path: String) virtual
void	_write_end() virtual
Error	_write_frame(frame_image: Image, audio_frame_block: `const void*`) virtual
void	add_writer(writer: MovieWriter) static

Method Descriptions

int **_get_audio_mix_rate**() virtual const 🔗

Called when the audio sample rate used for recording the audio is requested by the engine. The value returned must be specified in Hz. Defaults to 48000 Hz if _get_audio_mix_rate is not overridden.

SpeakerMode **_get_audio_speaker_mode**() virtual const 🔗

Called when the audio speaker mode used for recording the audio is requested by the engine. This can affect the number of output channels in the resulting audio file/stream. Defaults to AudioServer.SPEAKER_MODE_STEREO if _get_audio_speaker_mode is not overridden.

bool **_handles_file**(path: String) virtual const 🔗

Called when the engine determines whether this **MovieWriter** is able to handle the file at `path`. Must return `true` if this **MovieWriter** is able to handle the given file path, `false` otherwise. Typically, _handles_file is overridden as follows to allow the user to record a file at any path with a given file extension:

```
func _handles_file(path):
    # Allows specifying an output file with a `.mkv` file
extension (case-insensitive),
    # either in the Project Settings or with the `--write-movie
<path>` command line argument.
    return path.get_extension().to_lower() == "mkv"
```

Error **_write_begin**(movie_size: Vector2i, fps: int, base_path: String) virtual

Called once before the engine starts writing video and audio data. `movie_size` is the width and height of the video to save. `fps` is the number of frames per second specified in the project settings or using the `--fixed-fps <fps>` command line argument.

---

void **_write_end**() virtual

Called when the engine finishes writing. This occurs when the engine quits by pressing the window manager's close button, or when SceneTree.quit is called.

Note: Pressing `Ctrl + c` on the terminal running the editor/project does *not* result in _write_end being called.

Error **_write_frame**(frame_image: Image, audio_frame_block: `const void*`) virtual

Called at the end of every rendered frame. The `frame_image` and `audio_frame_block` function arguments should be written to.

---

void **add_writer**(writer: MovieWriter) static 🔗

Adds a writer to be usable by the engine. The supported file extensions can be set by overriding _handles_file.

Note: add_writer must be called early enough in the engine initialization to work, as movie writing is designed to start at the same time as the rest of the engine.

MultiplayerAPI

Inherits: RefCounted **<** Object

Inherited By: MultiplayerAPIExtension, SceneMultiplayer

High-level multiplayer API interface.

Description

Base class for high-level multiplayer API implementations. See also MultiplayerPeer.

By default, SceneTree has a reference to an implementation of this class and uses it to provide multiplayer capabilities (i.e. RPCs) across the whole scene.

It is possible to override the MultiplayerAPI instance used by specific tree branches by calling the SceneTree.set_multiplayer method, effectively allowing to run both client and server in the same scene.

It is also possible to extend or replace the default implementation via scripting or native extensions. See MultiplayerAPIExtension for details about extensions, SceneMultiplayer for the details about the default implementation.

Properties

MultiplayerPeer	multiplayer_peer

Methods

MultiplayerAPI	create_default_interface() static
StringName	get_default_interface() static
PackedInt32Array	get_peers()
int	get_remote_sender_id()
int	get_unique_id()
bool	has_multiplayer_peer()
bool	is_server()
Error	object_configuration_add(object: Object, configuration: Variant)
Error	object_configuration_remove(object: Object, configuration: Variant)
Error	poll()
Error	rpc(peer: int, object: Object, method: StringName, arguments: Array = [])

void	set_default_interface(interface_name: StringName) static

Signals

connected_to_server() 🔗

Emitted when this MultiplayerAPI's multiplayer_peer successfully connected to a server. Only emitted on clients.

connection_failed() 🔗

Emitted when this MultiplayerAPI's multiplayer_peer fails to establish a connection to a server. Only emitted on clients.

peer_connected(id: int) 🔗

Emitted when this MultiplayerAPI's multiplayer_peer connects with a new peer. ID is the peer ID of the new peer. Clients get notified when other clients connect to the same server. Upon connecting to a server, a client also receives this signal for the server (with ID being 1).

peer_disconnected(id: int) 🔗

Emitted when this MultiplayerAPI's multiplayer_peer disconnects from a peer. Clients get notified when other clients disconnect from the same server.

server_disconnected()

Emitted when this MultiplayerAPI's multiplayer_peer disconnects from server. Only emitted on clients.

Enumerations

enum **RPCMode**:

RPCMode **RPC_MODE_DISABLED** = `0`

Used with Node.rpc_config to disable a method or property for all RPC calls, making it unavailable. Default for all methods.

RPCMode **RPC_MODE_ANY_PEER** = `1`

Used with Node.rpc_config to set a method to be callable remotely by any peer. Analogous to the `@rpc("any_peer")` annotation. Calls are accepted from all remote peers, no matter if they are node's authority or not.

RPCMode **RPC_MODE_AUTHORITY** = `2`

Used with Node.rpc_config to set a method to be callable remotely only by the current multiplayer authority (which is the server by default). Analogous to the `@rpc("authority")` annotation. See Node.set_multiplayer_authority.

Property Descriptions

MultiplayerPeer **multiplayer_peer**

- void **set_multiplayer_peer**(value: MultiplayerPeer)

- MultiplayerPeer **get_multiplayer_peer**()

The peer object to handle the RPC system (effectively enabling networking when set). Depending on the peer itself, the MultiplayerAPI will become a network server (check with is_server) and will set root node's network mode to authority, or it will become a regular client peer. All child nodes are set to inherit the network mode by default. Handling of networking-related events (connection, disconnection, new clients) is done by connecting to MultiplayerAPI's signals.

Method Descriptions

MultiplayerAPI **create_default_interface**() static 🔗

Returns a new instance of the default MultiplayerAPI.

StringName **get_default_interface**() static 🔗

Returns the default MultiplayerAPI implementation class name. This is usually `"SceneMultiplayer"` when SceneMultiplayer is available. See set_default_interface.

PackedInt32Array **get_peers**() 🔗

Returns the peer IDs of all connected peers of this MultiplayerAPI's multiplayer_peer.

int **get_remote_sender_id**() 🔗

Returns the sender's peer ID for the RPC currently being executed.

Note: This method returns `0` when called outside of an RPC. As such, the original peer ID may be lost when code execution is delayed (such as with GDScript's `await` keyword).

---

int **get_unique_id**()

Returns the unique peer ID of this MultiplayerAPI's multiplayer_peer.

---

bool **has_multiplayer_peer**()

Returns `true` if there is a multiplayer_peer set.

---

bool **is_server**()

Returns `true` if this MultiplayerAPI's multiplayer_peer is valid and in server mode (listening for connections).

Error **object_configuration_add**(object: Object, configuration: Variant)

Notifies the MultiplayerAPI of a new `configuration` for the given `object`. This method is used internally by SceneTree to configure the root path for this MultiplayerAPI (passing `null` and a valid NodePath as `configuration`). This method can be further used by MultiplayerAPI implementations to provide additional features, refer to specific implementation (e.g. SceneMultiplayer) for details on how they use it.

Note: This method is mostly relevant when extending or overriding the MultiplayerAPI behavior via MultiplayerAPIExtension.

Error **object_configuration_remove**(object: Object, configuration: Variant) 🔗

Notifies the MultiplayerAPI to remove a `configuration` for the given `object`. This method is used internally by SceneTree to configure the root path for this MultiplayerAPI (passing `null` and an empty NodePath as `configuration`). This method can be further used by MultiplayerAPI implementations to provide additional features, refer to specific implementation (e.g. SceneMultiplayer) for details on how they use it.

Note: This method is mostly relevant when extending or overriding the MultiplayerAPI behavior via MultiplayerAPIExtension.

Error **poll**() 🔗

Method used for polling the MultiplayerAPI. You only need to worry about this if you set SceneTree.multiplayer_poll to `false`. By default, SceneTree will poll its MultiplayerAPI(s) for you.

Note: This method results in RPCs being called, so they will be executed in the same context of this function (e.g. `_process`, `physics`, Thread).

Error **rpc**(peer: int, object: Object, method: StringName, arguments: Array = []) 🔗

Sends an RPC to the target `peer`. The given `method` will be called on the remote `object` with the provided `arguments`. The RPC may also be called locally depending on the implementation and RPC configuration. See Node.rpc and Node.rpc_config.

Note: Prefer using Node.rpc, Node.rpc_id, or `my_method.rpc(peer, arg1, arg2, ...)` (in GDScript), since they are faster. This method is mostly useful in conjunction with MultiplayerAPIExtension when augmenting or replacing the multiplayer capabilities.

---

void **set_default_interface**(interface_name: StringName) static 🔗

Sets the default MultiplayerAPI implementation class. This method can be used by modules and extensions to configure which implementation will be used by SceneTree when the engine starts.

MultiplayerAPIExtension

Inherits: MultiplayerAPI **<** RefCounted **<** Object

Base class used for extending the MultiplayerAPI.

Description

This class can be used to augment or replace the default MultiplayerAPI implementation via script or extensions.

The following example augment the default implementation (SceneMultiplayer) by logging every RPC being made, and every object being configured for replication.

GDScript

```
extends MultiplayerAPIExtension
class_name LogMultiplayer

# We want to augment the default SceneMultiplayer.
var base_multiplayer = SceneMultiplayer.new()

func _init():
    # Just passthrough base signals (copied to var to avoid cyclic reference)
    var cts = connected_to_server
    var cf = connection_failed
    var pc = peer_connected
    var pd = peer_disconnected
    base_multiplayer.connected_to_server.connect(func(): cts.emit())
    base_multiplayer.connection_failed.connect(func(): cf.emit())
    base_multiplayer.peer_connected.connect(func(id): pc.emit(id))
    base_multiplayer.peer_disconnected.connect(func(id):
pd.emit(id))

func _poll():
    return base_multiplayer.poll()

# Log RPC being made and forward it to the default multiplayer.
func _rpc(peer: int, object: Object, method: StringName, args:
Array) -> Error:
    print("Got RPC for %d: %s::%s(%s)" % [peer, object, method,
args])
    return base_multiplayer.rpc(peer, object, method, args)

# Log configuration add. E.g. root path (nullptr, NodePath),
replication (Node, Spawner|Synchronizer), custom.
func _object_configuration_add(object, config: Variant) ->
Error:
    if config is MultiplayerSynchronizer:
        print("Adding synchronization configuration for %s.
Synchronizer: %s" % [object, config])
    elif config is MultiplayerSpawner:
        print("Adding node %s to the spawn list. Spawner: %s" %
[object, config])
    return base_multiplayer.object_configuration_add(object,
config)

# Log configuration remove. E.g. root path (nullptr, NodePath),
replication (Node, Spawner|Synchronizer), custom.
func _object_configuration_remove(object, config: Variant) ->
Error:
    if config is MultiplayerSynchronizer:
        print("Removing synchronization configuration for %s.
Synchronizer: %s" % [object, config])
    elif config is MultiplayerSpawner:
        print("Removing node %s from the spawn list. Spawner:
%s" % [object, config])
    return base_multiplayer.object_configuration_remove(object,
config)

# These can be optional, but in our case we want to augment
SceneMultiplayer, so forward everything.
func _set_multiplayer_peer(p_peer: MultiplayerPeer):
    base_multiplayer.multiplayer_peer = p_peer
```

```
func _get_multiplayer_peer() -> MultiplayerPeer:
    return base_multiplayer.multiplayer_peer

func _get_unique_id() -> int:
    return base_multiplayer.get_unique_id()

func _get_peer_ids() -> PackedInt32Array:
    return base_multiplayer.get_peers()
```

Then in your main scene or in an autoload call SceneTree.set_multiplayer to start using your custom MultiplayerAPI:

GDScript

```
# autoload.gd
func _enter_tree():
    # Sets our custom multiplayer as the main one in SceneTree.
get_tree().set_multiplayer(LogMultiplayer.new())
```

Native extensions can alternatively use the MultiplayerAPI.set_default_interface method during initialization to configure themselves as the default implementation.

Methods

MultiplayerPeer	_get_multiplayer_peer() virtual
PackedInt32Array	_get_peer_ids() virtual const
int	_get_remote_sender_id() virtual const
int	_get_unique_id() virtual const

Error	_object_configuration_add(object: Object, configuration: Variant) virtual
Error	_object_configuration_remove(object: Object, configuration: Variant) virtual
Error	_poll() virtual
Error	_rpc(peer: int, object: Object, method: StringName, args: Array) virtual
void	_set_multiplayer_peer(multiplayer_peer: MultiplayerPeer) virtual

Method Descriptions

MultiplayerPeer **_get_multiplayer_peer**() virtual 🔗

Called when the MultiplayerAPI.multiplayer_peer is retrieved.

PackedInt32Array **_get_peer_ids**() virtual const 🔗

Callback for MultiplayerAPI.get_peers.

int **_get_remote_sender_id**() virtual const 🔗

Callback for MultiplayerAPI.get_remote_sender_id.

---

int **_get_unique_id**() virtual const

Callback for MultiplayerAPI.get_unique_id.

Error **_object_configuration_add**(object: Object, configuration: Variant) virtual

Callback for MultiplayerAPI.object_configuration_add.

Error **_object_configuration_remove**(object: Object, configuration: Variant) virtual

Callback for MultiplayerAPI.object_configuration_remove.

Error **_poll**() virtual

Callback for MultiplayerAPI.poll.

Error **_rpc**(peer: int, object: Object, method: StringName, args: Array) virtual

Callback for MultiplayerAPI.rpc.

---

void **_set_multiplayer_peer**(multiplayer_peer: MultiplayerPeer) virtual

Called when the MultiplayerAPI.multiplayer_peer is set.

MultiplayerPeer

Inherits: PacketPeer **<** RefCounted **<** Object

Inherited By: ENetMultiplayerPeer, MultiplayerPeerExtension, OfflineMultiplayerPeer, WebRTCMultiplayerPeer, WebSocketMultiplayerPeer

Abstract class for specialized PacketPeers used by the MultiplayerAPI.

Description

Manages the connection with one or more remote peers acting as server or client and assigning unique IDs to each of them. See also MultiplayerAPI.

Note: The MultiplayerAPI protocol is an implementation detail and isn't meant to be used by non-Godot servers. It may change without notice.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Tutorials

- High-level multiplayer

Properties

bool	refuse_new_connections	`false`
int	transfer_channel	`0`
TransferMode	transfer_mode	`2`

Methods

void	close()
void	disconnect_peer(peer: int, force: bool = false)
int	generate_unique_id() const
ConnectionStatus	get_connection_status() const
int	get_packet_channel() const
TransferMode	get_packet_mode() const
int	get_packet_peer() const
int	get_unique_id() const

bool	is_server_relay_supported() const
void	poll()
void	set_target_peer(id: int)

Signals

peer_connected(id: int) 🔗

Emitted when a remote peer connects.

peer_disconnected(id: int) 🔗

Emitted when a remote peer has disconnected.

Enumerations

enum **ConnectionStatus**: 🔗

ConnectionStatus **CONNECTION_DISCONNECTED** = `0`

The MultiplayerPeer is disconnected.

ConnectionStatus **CONNECTION_CONNECTING** = `1`

The MultiplayerPeer is currently connecting to a server.

ConnectionStatus **CONNECTION_CONNECTED** = `2`

This MultiplayerPeer is connected.

enum **TransferMode**: 🔗

TransferMode **TRANSFER_MODE_UNRELIABLE** = `0`

Packets are not acknowledged, no resend attempts are made for lost packets. Packets may arrive in any order. Potentially faster than TRANSFER_MODE_UNRELIABLE_ORDERED. Use for non-critical data, and always consider whether the order matters.

TransferMode **TRANSFER_MODE_UNRELIABLE_ORDERED** = `1`

Packets are not acknowledged, no resend attempts are made for lost packets. Packets are received in the order they were sent in. Potentially faster than TRANSFER_MODE_RELIABLE. Use for non-critical data or data that would be outdated if received late due to resend attempt(s) anyway, for example movement and positional data.

TransferMode **TRANSFER_MODE_RELIABLE** = `2`

Packets must be received and resend attempts should be made until the packets are acknowledged. Packets must be received in the order they were sent in. Most reliable transfer mode, but potentially the slowest due to the overhead. Use for critical data that must be transmitted and arrive in order, for example an ability being triggered or a chat message. Consider carefully if the information really is critical, and use sparingly.

Constants

TARGET_PEER_BROADCAST = `0` 🔗

Packets are sent to all connected peers.

TARGET_PEER_SERVER = `1` 🔗

Packets are sent to the remote peer acting as server.

Property Descriptions

bool **refuse_new_connections** = `false` 🔗

- void **set_refuse_new_connections**(value: bool)
- bool **is_refusing_new_connections**()

If `true`, this **MultiplayerPeer** refuses new connections.

---

int **transfer_channel** = `0` 🔗

- void **set_transfer_channel**(value: int)
- int **get_transfer_channel**()

The channel to use to send packets. Many network APIs such as ENet and WebRTC allow the creation of multiple independent channels which behaves, in a way, like separate connections. This means that reliable data will only block delivery of other packets on that channel, and ordering will only be in respect to the channel the packet is being sent on. Using different channels to send **different and independent** state updates is a common way to optimize network usage and decrease latency in fast-paced games.

Note: The default channel (`0`) actually works as 3 separate channels (one for each TransferMode) so that TRANSFER_MODE_RELIABLE and TRANSFER_MODE_UNRELIABLE_ORDERED does not interact with each other by default. Refer to the specific network API documentation (e.g. ENet or WebRTC) to learn how to set up channels correctly.

TransferMode **transfer_mode** = `2` 🔗

- void **set_transfer_mode**(value: TransferMode)
- TransferMode **get_transfer_mode**()

The manner in which to send packets to the target peer. See TransferMode, and the set_target_peer method.

Method Descriptions

void **close**() 🔗

Immediately close the multiplayer peer returning to the state CONNECTION_DISCONNECTED. Connected peers will be dropped without emitting peer_disconnected.

---

void **disconnect_peer**(peer: int, force: bool = false) 🔗

Disconnects the given `peer` from this host. If `force` is `true` the peer_disconnected signal will not be emitted for this peer.

---

int **generate_unique_id**() const 🔗

Returns a randomly generated integer that can be used as a network unique ID.

ConnectionStatus **get_connection_status**() const 🔗

Returns the current state of the connection. See ConnectionStatus.

---

int **get_packet_channel**() const

Returns the channel over which the next available packet was received. See PacketPeer.get_available_packet_count.

TransferMode **get_packet_mode**() const

Returns the transfer mode the remote peer used to send the next available packet. See PacketPeer.get_available_packet_count.

---

int **get_packet_peer**() const

Returns the ID of the **MultiplayerPeer** who sent the next available packet. See PacketPeer.get_available_packet_count.

---

int **get_unique_id**() const

Returns the ID of this **MultiplayerPeer**.

---

bool **is_server_relay_supported**() const

Returns true if the server can act as a relay in the current configuration (i.e. if the higher level MultiplayerAPI should notify connected clients of other peers, and implement a relay protocol to allow communication between them).

---

void **poll**()

Waits up to 1 second to receive a new network event.

---

void **set_target_peer**(id: int)

Sets the peer to which packets will be sent.

The `id` can be one of: TARGET_PEER_BROADCAST to send to all connected peers, TARGET_PEER_SERVER to send to the peer acting as server, a valid peer ID to send to that specific peer, a negative peer ID to send to all peers except that one. By default, the target peer is TARGET_PEER_BROADCAST.

MultiplayerPeerExtension

Inherits: MultiplayerPeer **<** PacketPeer **<** RefCounted **<** Object

Class that can be inherited to implement custom multiplayer API networking layers via GDExtension.

Description

This class is designed to be inherited from a GDExtension plugin to implement custom networking layers for the multiplayer API (such as WebRTC). All the methods below **must** be implemented to have a working custom multiplayer implementation. See also MultiplayerAPI.

Methods

void	_close() virtual
void	_disconnect_peer(p_peer: int, p_force: bool) virtual
int	_get_available_packet_count() virtual const
ConnectionStatus	_get_connection_status() virtual const
int	_get_max_packet_size() virtual const

Error	_get_packet(r_buffer: `const uint8_t **`, r_buffer_size: `int32_t*`) virtual
int	_get_packet_channel() virtual const
TransferMode	_get_packet_mode() virtual const
int	_get_packet_peer() virtual const
PackedByteArray	_get_packet_script() virtual
int	_get_transfer_channel() virtual const
TransferMode	_get_transfer_mode() virtual const
int	_get_unique_id() virtual const
bool	_is_refusing_new_connections() virtual const
bool	_is_server() virtual const
bool	_is_server_relay_supported() virtual const
void	_poll() virtual

Error	_put_packet(p_buffer: `const uint8_t*`, p_buffer_size: int) virtual
Error	_put_packet_script(p_buffer: PackedByteArray) virtual
void	_set_refuse_new_connections(p_enable: bool) virtual
void	_set_target_peer(p_peer: int) virtual
void	_set_transfer_channel(p_channel: int) virtual
void	_set_transfer_mode(p_mode: TransferMode) virtual

Method Descriptions

void **_close**() virtual

Called when the multiplayer peer should be immediately closed (see MultiplayerPeer.close).

---

void **_disconnect_peer**(p_peer: int, p_force: bool) virtual

Called when the connected `p_peer` should be forcibly disconnected (see MultiplayerPeer.disconnect_peer).

---

int **_get_available_packet_count**() virtual const 🔗

Called when the available packet count is internally requested by the MultiplayerAPI.

ConnectionStatus **_get_connection_status**() virtual const 🔗

Called when the connection status is requested on the MultiplayerPeer (see MultiplayerPeer.get_connection_status).

---

int **_get_max_packet_size**() virtual const 🔗

Called when the maximum allowed packet size (in bytes) is requested by the MultiplayerAPI.

Error **_get_packet**(r_buffer: `const uint8_t **`, r_buffer_size: `int32_t*`) virtual 🔗

Called when a packet needs to be received by the MultiplayerAPI, with `r_buffer_size` being the size of the binary `r_buffer` in bytes.

---

int **_get_packet_channel**() virtual const 🔗

Called to get the channel over which the next available packet was received. See MultiplayerPeer.get_packet_channel.

TransferMode **_get_packet_mode**() virtual const 🔗

Called to get the transfer mode the remote peer used to send the next available packet. See MultiplayerPeer.get_packet_mode.

---

int **_get_packet_peer**() virtual const 🔗

Called when the ID of the MultiplayerPeer who sent the most recent packet is requested (see MultiplayerPeer.get_packet_peer).

PackedByteArray **_get_packet_script**() virtual 🔗

Called when a packet needs to be received by the MultiplayerAPI, if _get_packet isn't implemented. Use this when extending this class via GDScript.

---

int **_get_transfer_channel**() virtual const 🔗

Called when the transfer channel to use is read on this MultiplayerPeer (see MultiplayerPeer.transfer_channel).

TransferMode **_get_transfer_mode**() virtual const 🔗

Called when the transfer mode to use is read on this MultiplayerPeer (see MultiplayerPeer.transfer_mode).

---

int **_get_unique_id**() virtual const 🔗

Called when the unique ID of this MultiplayerPeer is requested (see MultiplayerPeer.get_unique_id). The value must be between `1` and `2147483647`.

---

bool **_is_refusing_new_connections**() virtual const 🔗

Called when the "refuse new connections" status is requested on this MultiplayerPeer (see MultiplayerPeer.refuse_new_connections).

---

bool **_is_server**() virtual const

Called when the "is server" status is requested on the MultiplayerAPI. See MultiplayerAPI.is_server.

bool **_is_server_relay_supported**() virtual const

Called to check if the server can act as a relay in the current configuration. See MultiplayerPeer.is_server_relay_supported.

void **_poll**() virtual

Called when the MultiplayerAPI is polled. See MultiplayerAPI.poll.

Error **_put_packet**(p_buffer: `const uint8_t*`, p_buffer_size: int) virtual

Called when a packet needs to be sent by the MultiplayerAPI, with `p_buffer_size` being the size of the binary `p_buffer` in bytes.

Error **_put_packet_script**(p_buffer: PackedByteArray) virtual

Called when a packet needs to be sent by the MultiplayerAPI, if _put_packet isn't implemented. Use this when extending this class via GDScript.

void **_set_refuse_new_connections**(p_enable: bool) virtual

Called when the "refuse new connections" status is set on this MultiplayerPeer (see MultiplayerPeer.refuse_new_connections).

void **_set_target_peer**(p_peer: int) virtual 🔗

Called when the target peer to use is set for this MultiplayerPeer (see MultiplayerPeer.set_target_peer).

void **_set_transfer_channel**(p_channel: int) virtual 🔗

Called when the channel to use is set for this MultiplayerPeer (see MultiplayerPeer.transfer_channel).

void **_set_transfer_mode**(p_mode: TransferMode) virtual 🔗

Called when the transfer mode is set on this MultiplayerPeer (see MultiplayerPeer.transfer_mode).

Mutex

Inherits: RefCounted **<** Object

A binary Semaphore for synchronization of multiple Threads.

Description

A synchronization mutex (mutual exclusion). This is used to synchronize multiple Threads, and is equivalent to a binary Semaphore. It guarantees that only one thread can access a critical section at a time.

This is a reentrant mutex, meaning that it can be locked multiple times by one thread, provided it also unlocks it as many times.

Warning: Mutexes must be used carefully to avoid deadlocks.

Warning: To ensure proper cleanup without crashes or deadlocks, the following conditions must be met:

- When a **Mutex**'s reference count reaches zero and it is therefore destroyed, no threads (including the one on which the destruction will happen) must have it locked.
- When a Thread's reference count reaches zero and it is therefore destroyed, it must not have any mutex locked.

Tutorials

- Using multiple threads
- Thread-safe APIs

Methods

void	lock()
bool	try_lock()
void	unlock()

Method Descriptions

void **lock()** 🔗

Locks this **Mutex**, blocks until it is unlocked by the current owner.

Note: This function returns without blocking if the thread already has ownership of the mutex.

---

bool **try_lock()** 🔗

Tries locking this **Mutex**, but does not block. Returns `true` on success, `false` otherwise.

Note: This function returns `true` if the thread already has ownership of the mutex.

---

void **unlock()** 🔗

Unlocks this **Mutex**, leaving it to other threads.

Note: If a thread called lock or try_lock multiple times while already having ownership of the mutex, it must also call unlock the same number of times in order to unlock it correctly.

Warning: Calling unlock more times that lock on a given thread, thus ending up trying to unlock a non-locked mutex, is wrong and may causes crashes or deadlocks.

NativeMenu

Inherits: Object

A server interface for OS native menus.

Description

NativeMenu handles low-level access to the OS native global menu bar and popup menus.

Note: This is low-level API, consider using MenuBar with MenuBar.prefer_global_menu set to `true`, and PopupMenu with PopupMenu.prefer_native_menu set to `true`.

To create a menu, use create_menu, add menu items using `add_*_item` methods. To remove a menu, use free_menu.

```
var menu

func _menu_callback(item_id):
    if item_id == "ITEM_CUT":
        cut()
    elif item_id == "ITEM_COPY":
        copy()
    elif item_id == "ITEM_PASTE":
        paste()

func _enter_tree():
    # Create new menu and add items:
    menu = NativeMenu.create_menu()
    NativeMenu.add_item(menu, "Cut", _menu_callback,
Callable(), "ITEM_CUT")
    NativeMenu.add_item(menu, "Copy", _menu_callback,
Callable(), "ITEM_COPY")
    NativeMenu.add_separator(menu)
    NativeMenu.add_item(menu, "Paste", _menu_callback,
Callable(), "ITEM_PASTE")

func _on_button_pressed():
    # Show popup menu at mouse position:
    NativeMenu.popup(menu, DisplayServer.mouse_get_position())

func _exit_tree():
    # Remove menu when it's no longer needed:
    NativeMenu.free_menu(menu)
```

Methods

int	add_check_item(rid: RID, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	add_icon_check_item(rid: RID, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	add_icon_item(rid: RID, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	add_icon_radio_check_item(rid: RID, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)

int	add_item(rid: RID, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	add_multistate_item(rid: RID, label: String, max_states: int, default_state: int, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	add_radio_check_item(rid: RID, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)
int	add_separator(rid: RID, index: int = -1)
int	add_submenu_item(rid: RID, label: String, submenu_rid: RID, tag: Variant = null, index: int = -1)
void	clear(rid: RID)
RID	create_menu()
int	find_item_index_with_submenu(rid: RID, submenu_rid: RID) const
int	find_item_index_with_tag(rid: RID, tag: Variant) const

int	find_item_index_with_text(rid: RID, text: String) const
void	free_menu(rid: RID)
Key	get_item_accelerator(rid: RID, idx: int) const
Callable	get_item_callback(rid: RID, idx: int) const
int	get_item_count(rid: RID) const
Texture2D	get_item_icon(rid: RID, idx: int) const
int	get_item_indentation_level(rid: RID, idx: int) const
Callable	get_item_key_callback(rid: RID, idx: int) const
int	get_item_max_states(rid: RID, idx: int) const
int	get_item_state(rid: RID, idx: int) const
RID	get_item_submenu(rid: RID, idx: int) const
Variant	get_item_tag(rid: RID, idx: int) const
String	get_item_text(rid: RID, idx: int) const

String	get_item_tooltip(rid: RID, idx: int) const
float	get_minimum_width(rid: RID) const
Callable	get_popup_close_callback(rid: RID) const
Callable	get_popup_open_callback(rid: RID) const
Vector2	get_size(rid: RID) const
RID	get_system_menu(menu_id: SystemMenus) const
String	get_system_menu_name(menu_id: SystemMenus) const
bool	has_feature(feature: Feature) const
bool	has_menu(rid: RID) const
bool	has_system_menu(menu_id: SystemMenus) const
bool	is_item_checkable(rid: RID, idx: int) const
bool	is_item_checked(rid: RID, idx: int) const

bool	is_item_disabled(rid: RID, idx: int) const
bool	is_item_hidden(rid: RID, idx: int) const
bool	is_item_radio_checkable(rid: RID, idx: int) const
bool	is_opened(rid: RID) const
bool	is_system_menu(rid: RID) const
void	popup(rid: RID, position: Vector2i)
void	remove_item(rid: RID, idx: int)
void	set_interface_direction(rid: RID, is_rtl: bool)
void	set_item_accelerator(rid: RID, idx: int, keycode: Key)
void	set_item_callback(rid: RID, idx: int, callback: Callable)
void	set_item_checkable(rid: RID, idx: int, checkable: bool)
void	set_item_checked(rid: RID, idx: int, checked: bool)
void	set_item_disabled(rid: RID, idx: int, disabled: bool)

void	set_item_hidden(rid: RID, idx: int, hidden: bool)
void	set_item_hover_callbacks(rid: RID, idx: int, callback: Callable)
void	set_item_icon(rid: RID, idx: int, icon: Texture2D)
void	set_item_indentation_level(rid: RID, idx: int, level: int)
void	set_item_key_callback(rid: RID, idx: int, key_callback: Callable)
void	set_item_max_states(rid: RID, idx: int, max_states: int)
void	set_item_radio_checkable(rid: RID, idx: int, checkable: bool)
void	set_item_state(rid: RID, idx: int, state: int)
void	set_item_submenu(rid: RID, idx: int, submenu_rid: RID)
void	set_item_tag(rid: RID, idx: int, tag: Variant)
void	set_item_text(rid: RID, idx: int, text: String)

void	set_item_tooltip(rid: RID, idx: int, tooltip: String)
void	set_minimum_width(rid: RID, width: float)
void	set_popup_close_callback(rid: RID, callback: Callable)
void	set_popup_open_callback(rid: RID, callback: Callable)

Enumerations

enum **Feature**: 🔗

Feature **FEATURE_GLOBAL_MENU** = `0`

NativeMenu supports native global main menu.

Feature **FEATURE_POPUP_MENU** = `1`

NativeMenu supports native popup menus.

Feature **FEATURE_OPEN_CLOSE_CALLBACK** = `2`

NativeMenu supports menu open and close callbacks.

Feature **FEATURE_HOVER_CALLBACK** = `3`

NativeMenu supports menu item hover callback.

Feature **FEATURE_KEY_CALLBACK** = `4`

NativeMenu supports menu item accelerator/key callback.

---

enum **SystemMenus**: 🔗

SystemMenus **INVALID_MENU_ID** = `0`

Invalid special system menu ID.

SystemMenus **MAIN_MENU_ID** = `1`

Global main menu ID.

SystemMenus **APPLICATION_MENU_ID** = `2`

Application (first menu after "Apple" menu on macOS) menu ID.

SystemMenus **WINDOW_MENU_ID** = `3`

"Window" menu ID (on macOS this menu includes standard window control items and a list of open windows).

SystemMenus **HELP_MENU_ID** = `4`

"Help" menu ID (on macOS this menu includes help search bar).

SystemMenus **DOCK_MENU_ID** = `5`

Dock icon right-click menu ID (on macOS this menu include standard application control items and a list of open windows).

Method Descriptions

int **add_check_item**(rid: RID, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1) 🔗

Adds a new checkable item with text `label` to the global menu `rid`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented on macOS and Windows.

Note: On Windows, `accelerator` and `key_callback` are ignored.

---

int **add_icon_check_item**(rid: RID, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)

Adds a new checkable item with text `label` and icon `icon` to the global menu `rid`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented on macOS and Windows.

Note: On Windows, `accelerator` and `key_callback` are ignored.

---

int **add_icon_item**(rid: RID, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)

Adds a new item with text `label` and icon `icon` to the global menu `rid`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented on macOS and Windows.

Note: On Windows, `accelerator` and `key_callback` are ignored.

---

int **add_icon_radio_check_item**(rid: RID, icon: Texture2D, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1)

Adds a new radio-checkable item with text `label` and icon `icon` to the global menu `rid`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: Radio-checkable items just display a checkmark, but don't have any built-in checking behavior and must be checked/unchecked manually. See set_item_checked for more info on how to control it.

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented on macOS and Windows.

Note: On Windows, `accelerator` and `key_callback` are ignored.

---

int **add_item**(rid: RID, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1) 🔗

Adds a new item with text `label` to the global menu `rid`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented on macOS and Windows.

Note: On Windows, `accelerator` and `key_callback` are ignored.

---

int **add_multistate_item**(rid: RID, label: String, max_states: int, default_state: int, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1) 🔗

Adds a new item with text `label` to the global menu `rid`.

Contrarily to normal binary items, multistate items can have more than two states, as defined by `max_states`. Each press or activate of the item will increase the state by one. The default value is defined by `default_state`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: By default, there's no indication of the current item state, it should be changed manually.

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented on macOS and Windows.

Note: On Windows, `accelerator` and `key_callback` are ignored.

---

int **add_radio_check_item**(rid: RID, label: String, callback: Callable = Callable(), key_callback: Callable = Callable(), tag: Variant = null, accelerator: Key = 0, index: int = -1) 🔗

Adds a new radio-checkable item with text `label` to the global menu `rid`.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

An `accelerator` can optionally be defined, which is a keyboard shortcut that can be pressed to trigger the menu button even if it's not currently open. The `accelerator` is generally a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: Radio-checkable items just display a checkmark, but don't have any built-in checking behavior and must be checked/unchecked manually. See set_item_checked for more info on how to control it.

Note: The `callback` and `key_callback` Callables need to accept exactly one Variant parameter, the parameter passed to the Callables will be the value passed to `tag`.

Note: This method is implemented on macOS and Windows.

Note: On Windows, `accelerator` and `key_callback` are ignored.

---

int **add_separator**(rid: RID, index: int = -1) 🔗

Adds a separator between items to the global menu `rid`. Separators also occupy an index.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

Note: This method is implemented on macOS and Windows.

---

int **add_submenu_item**(rid: RID, label: String, submenu_rid: RID, tag: Variant = null, index: int = -1) 🔗

Adds an item that will act as a submenu of the global menu `rid`. The `submenu_rid` argument is the RID of the global menu that will be shown when the item is clicked.

Returns index of the inserted item, it's not guaranteed to be the same as `index` value.

Note: This method is implemented on macOS and Windows.

---

void **clear**(rid: RID) 🔗

Removes all items from the global menu `rid`.

Note: This method is implemented on macOS and Windows.

RID **create_menu**() 🔗

Creates a new global menu object.

Note: This method is implemented on macOS and Windows.

---

int **find_item_index_with_submenu**(rid: RID, submenu_rid: RID) const 🔗

Returns the index of the item with the submenu specified by `submenu_rid`. Indices are automatically assigned to each item by the engine, and cannot be set manually.

Note: This method is implemented on macOS and Windows.

---

int **find_item_index_with_tag**(rid: RID, tag: Variant) const 🔗

Returns the index of the item with the specified `tag`. Indices are automatically assigned to each item by the engine, and cannot be set manually.

Note: This method is implemented on macOS and Windows.

---

int **find_item_index_with_text**(rid: RID, text: String) const 🔗

Returns the index of the item with the specified `text`. Indices are automatically assigned to each item by the engine, and cannot be set manually.

Note: This method is implemented on macOS and Windows.

---

void **free_menu**(rid: RID) 🔗

Frees a global menu object created by this **NativeMenu**.

Note: This method is implemented on macOS and Windows.

Key **get_item_accelerator**(rid: RID, idx: int) const 🔗

Returns the accelerator of the item at index `idx`. Accelerators are special combinations of keys that activate the item, no matter which control is focused.

Note: This method is implemented only on macOS.

Callable **get_item_callback**(rid: RID, idx: int) const

Returns the callback of the item at index `idx`.

Note: This method is implemented on macOS and Windows.

---

int **get_item_count**(rid: RID) const

Returns number of items in the global menu `rid`.

Note: This method is implemented on macOS and Windows.

Texture2D **get_item_icon**(rid: RID, idx: int) const

Returns the icon of the item at index `idx`.

Note: This method is implemented on macOS and Windows.

---

int **get_item_indentation_level**(rid: RID, idx: int) const

Returns the horizontal offset of the item at the given `idx`.

Note: This method is implemented only on macOS.

Callable **get_item_key_callback**(rid: RID, idx: int) const

Returns the callback of the item accelerator at index `idx`.

Note: This method is implemented only on macOS.

---

int **get_item_max_states**(rid: RID, idx: int) const

Returns number of states of a multistate item. See add_multistate_item for details.

Note: This method is implemented on macOS and Windows.

---

int **get_item_state**(rid: RID, idx: int) const

Returns the state of a multistate item. See add_multistate_item for details.

Note: This method is implemented on macOS and Windows.

RID **get_item_submenu**(rid: RID, idx: int) const

Returns the submenu ID of the item at index `idx`. See add_submenu_item for more info on how to add a submenu.

Note: This method is implemented on macOS and Windows.

Variant **get_item_tag**(rid: RID, idx: int) const

Returns the metadata of the specified item, which might be of any type. You can set it with set_item_tag, which provides a simple way of assigning context data to items.

Note: This method is implemented on macOS and Windows.

String **get_item_text**(rid: RID, idx: int) const

Returns the text of the item at index `idx`.

Note: This method is implemented on macOS and Windows.

String **get_item_tooltip**(rid: RID, idx: int) const

Returns the tooltip associated with the specified index `idx`.

Note: This method is implemented only on macOS.

---

float **get_minimum_width**(rid: RID) const

Returns global menu minimum width.

Note: This method is implemented only on macOS.

Callable **get_popup_close_callback**(rid: RID) const

Returns global menu close callback.

b]Note:** This method is implemented only on macOS.

Callable **get_popup_open_callback**(rid: RID) const

Returns global menu open callback.

b]Note:** This method is implemented only on macOS.

Vector2 **get_size**(rid: RID) const

Returns global menu size.

Note: This method is implemented on macOS and Windows.

RID **get_system_menu**(menu_id: SystemMenus) const

Returns RID of a special system menu.

Note: This method is implemented only on macOS.

String **get_system_menu_name**(menu_id: SystemMenus) const

Returns readable name of a special system menu.

Note: This method is implemented only on macOS.

---

bool **has_feature**(feature: Feature) const

Returns `true` if the specified `feature` is supported by the current **NativeMenu**, `false` otherwise.

Note: This method is implemented on macOS and Windows.

---

bool **has_menu**(rid: RID) const

Returns `true` if `rid` is valid global menu.

Note: This method is implemented on macOS and Windows.

---

bool **has_system_menu**(menu_id: SystemMenus) const

Returns `true` if a special system menu is supported.

Note: This method is implemented only on macOS.

---

bool **is_item_checkable**(rid: RID, idx: int) const

Returns `true` if the item at index `idx` is checkable in some way, i.e. if it has a checkbox or radio button.

Note: This method is implemented on macOS and Windows.

---

bool **is_item_checked**(rid: RID, idx: int) const

Returns `true` if the item at index `idx` is checked.

Note: This method is implemented on macOS and Windows.

---

bool **is_item_disabled**(rid: RID, idx: int) const

Returns `true` if the item at index `idx` is disabled. When it is disabled it can't be selected, or its action invoked.

See set_item_disabled for more info on how to disable an item.

Note: This method is implemented on macOS and Windows.

---

bool **is_item_hidden**(rid: RID, idx: int) const

Returns `true` if the item at index `idx` is hidden.

See set_item_hidden for more info on how to hide an item.

Note: This method is implemented only on macOS.

---

bool **is_item_radio_checkable**(rid: RID, idx: int) const 🔗

Returns `true` if the item at index `idx` has radio button-style checkability.

Note: This is purely cosmetic; you must add the logic for checking/unchecking items in radio groups.

Note: This method is implemented on macOS and Windows.

---

bool **is_opened**(rid: RID) const 🔗

Returns `true` if the menu is currently opened.

Note: This method is implemented only on macOS.

---

bool **is_system_menu**(rid: RID) const 🔗

Return `true` is global menu is a special system menu.

Note: This method is implemented only on macOS.

---

void **popup**(rid: RID, position: Vector2i) 🔗

Shows the global menu at `position` in the screen coordinates.

Note: This method is implemented on macOS and Windows.

---

void **remove_item**(rid: RID, idx: int) 🔗

Removes the item at index `idx` from the global menu `rid`.

Note: The indices of items after the removed item will be shifted by one.

Note: This method is implemented on macOS and Windows.

void **set_interface_direction**(rid: RID, is_rtl: bool)

Sets the menu text layout direction from right-to-left if `is_rtl` is `true`.

Note: This method is implemented on macOS and Windows.

void **set_item_accelerator**(rid: RID, idx: int, keycode: Key)

Sets the accelerator of the item at index `idx`. `keycode` can be a single Key, or a combination of KeyModifierMasks and Keys using bitwise OR such as `KEY_MASK_CTRL | KEY_A` (`Ctrl + A`).

Note: This method is implemented only on macOS.

void **set_item_callback**(rid: RID, idx: int, callback: Callable)

Sets the callback of the item at index `idx`. Callback is emitted when an item is pressed.

Note: The `callback` Callable needs to accept exactly one Variant parameter, the parameter passed to the Callable will be the value passed to the `tag` parameter when the menu item was created.

Note: This method is implemented on macOS and Windows.

void **set_item_checkable**(rid: RID, idx: int, checkable: bool)

Sets whether the item at index `idx` has a checkbox. If `false`, sets the type of the item to plain text.

Note: This method is implemented on macOS and Windows.

---

void **set_item_checked**(rid: RID, idx: int, checked: bool)

Sets the checkstate status of the item at index `idx`.

Note: This method is implemented on macOS and Windows.

---

void **set_item_disabled**(rid: RID, idx: int, disabled: bool)

Enables/disables the item at index `idx`. When it is disabled, it can't be selected and its action can't be invoked.

Note: This method is implemented on macOS and Windows.

---

void **set_item_hidden**(rid: RID, idx: int, hidden: bool)

Hides/shows the item at index `idx`. When it is hidden, an item does not appear in a menu and its action cannot be invoked.

Note: This method is implemented only on macOS.

---

void **set_item_hover_callbacks**(rid: RID, idx: int, callback: Callable)

Sets the callback of the item at index `idx`. The callback is emitted when an item is hovered.

Note: The `callback` Callable needs to accept exactly one Variant parameter, the parameter passed to the Callable will be the value passed to the `tag` parameter when the menu item was created.

Note: This method is implemented only on macOS.

---

void **set_item_icon**(rid: RID, idx: int, icon: Texture2D)

Replaces the Texture2D icon of the specified `idx`.

Note: This method is implemented on macOS and Windows.

Note: This method is not supported by macOS Dock menu items.

---

void **set_item_indentation_level**(rid: RID, idx: int, level: int)

Sets the horizontal offset of the item at the given `idx`.

Note: This method is implemented only on macOS.

---

void **set_item_key_callback**(rid: RID, idx: int, key_callback: Callable)

Sets the callback of the item at index `idx`. Callback is emitted when its accelerator is activated.

Note: The `key_callback` Callable needs to accept exactly one Variant parameter, the parameter passed to the Callable will be the value passed to the `tag` parameter when the menu item was created.

Note: This method is implemented only on macOS.

---

void **set_item_max_states**(rid: RID, idx: int, max_states: int)

Sets number of state of a multistate item. See add_multistate_item for details.

Note: This method is implemented on macOS and Windows.

---

void **set_item_radio_checkable**(rid: RID, idx: int, checkable: bool)

Sets the type of the item at the specified index `idx` to radio button. If `false`, sets the type of the item to plain text.

Note: This is purely cosmetic; you must add the logic for checking/unchecking items in radio groups.

Note: This method is implemented on macOS and Windows.

---

void **set_item_state**(rid: RID, idx: int, state: int)

Sets the state of a multistate item. See add_multistate_item for details.

Note: This method is implemented on macOS and Windows.

---

void **set_item_submenu**(rid: RID, idx: int, submenu_rid: RID)

Sets the submenu RID of the item at index `idx`. The submenu is a global menu that would be shown when the item is clicked.

Note: This method is implemented on macOS and Windows.

---

void **set_item_tag**(rid: RID, idx: int, tag: Variant)

Sets the metadata of an item, which may be of any type. You can later get it with get_item_tag, which provides a simple way of assigning context data to items.

Note: This method is implemented on macOS and Windows.

---

void **set_item_text**(rid: RID, idx: int, text: String) 🔗

Sets the text of the item at index `idx`.

Note: This method is implemented on macOS and Windows.

---

void **set_item_tooltip**(rid: RID, idx: int, tooltip: String) 🔗

Sets the String tooltip of the item at the specified index `idx`.

Note: This method is implemented only on macOS.

---

void **set_minimum_width**(rid: RID, width: float) 🔗

Sets the minimum width of the global menu.

Note: This method is implemented only on macOS.

---

void **set_popup_close_callback**(rid: RID, callback: Callable) 🔗

Registers callable to emit when the menu is about to show.

Note: The OS can simulate menu opening to track menu item changes and global shortcuts, in which case the corresponding close callback is not triggered. Use is_opened to check if the menu is currently opened.

Note: This method is implemented only on macOS.

---

void **set_popup_open_callback**(rid: RID, callback: Callable) 🔗

Registers callable to emit after the menu is closed.

Note: This method is implemented only on macOS.

NavigationMeshGenerator

Deprecated: This class may be changed or removed in future versions.

Inherits: Object

Helper class for creating and clearing navigation meshes.

Description

This class is responsible for creating and clearing 3D navigation meshes used as NavigationMesh resources inside NavigationRegion3D. The **NavigationMeshGenerator** has very limited to no use for 2D as the navigation mesh baking process expects 3D node types and 3D source geometry to parse.

The entire navigation mesh baking is best done in a separate thread as the voxelization, collision tests and mesh optimization steps involved are very slow and performance-intensive operations.

Navigation mesh baking happens in multiple steps and the result depends on 3D source geometry and properties of the NavigationMesh resource. In the first step, starting from a root node and depending on NavigationMesh properties all valid 3D source geometry nodes are collected from the SceneTree. Second, all collected nodes are parsed for their relevant 3D geometry data and a combined 3D mesh is build. Due to the many different types of parsable objects, from normal MeshInstance3Ds to CSGShape3Ds or various CollisionObject3Ds, some operations to collect geometry data can trigger RenderingServer and PhysicsServer3D synchronizations. Server synchronization can have a negative effect on baking time or framerate as it often involves Mutex locking for thread security. Many parsable objects and the continuous synchronization with other threaded Servers can increase the baking time significantly. On the other hand only a few but very large and complex objects will take some time to prepare for the Servers which can noticeably stall the next frame render. As a general rule the total number of parsable objects and their individual size and complexity should be balanced to avoid framerate issues or very long baking times. The combined mesh is then passed to the Recast Navigation Object to test the source geometry for walkable terrain suitable to NavigationMesh agent properties by creating a voxel world around the meshes bounding area.

The finalized navigation mesh is then returned and stored inside the NavigationMesh for use as a resource inside NavigationRegion3D nodes.

Note: Using meshes to not only define walkable surfaces but also obstruct navigation baking does not always work. The navigation baking has no concept of what is a geometry "inside" when dealing with mesh source geometry and this is intentional. Depending on current baking parameters, as soon as the obstructing mesh is large enough to fit a navigation mesh area inside, the baking will generate navigation mesh areas that are inside the obstructing source geometry mesh.

Tutorials

- Using NavigationMeshes

Methods

void bake(navigation_mesh: NavigationMesh, root_node: Node)

---

void	bake_from_source_geometry_data(navigation_mesh: NavigationMesh, source_geometry_data: NavigationMeshSourceGeometryData3D, callback: Callable = Callable())
void	clear(navigation_mesh: NavigationMesh)
void	parse_source_geometry_data(navigation_mesh: NavigationMesh, source_geometry_data: NavigationMeshSourceGeometryData3D, root_node: Node, callback: Callable = Callable())

Method Descriptions

void **bake**(navigation_mesh: NavigationMesh, root_node: Node) 🔗

Deprecated: This method is deprecated due to core threading changes. To upgrade existing code, first create a NavigationMeshSourceGeometryData3D resource. Use this resource with parse_source_geometry_data to parse the SceneTree for nodes that should contribute to the navigation mesh baking. The SceneTree parsing needs to happen on the main thread. After the parsing is finished use the resource with bake_from_source_geometry_data to bake a navigation mesh.

Bakes the `navigation_mesh` with source geometry collected starting from the `root_node`.

---

void **bake_from_source_geometry_data**(navigation_mesh: NavigationMesh, source_geometry_data:

NavigationMeshSourceGeometryData3D, callback: Callable = Callable()) 🔗

Bakes the provided `navigation_mesh` with the data from the provided `source_geometry_data`. After the process is finished the optional `callback` will be called.

---

void **clear**(navigation_mesh: NavigationMesh) 🔗

Removes all polygons and vertices from the provided `navigation_mesh` resource.

---

void **parse_source_geometry_data**(navigation_mesh: NavigationMesh, source_geometry_data: NavigationMeshSourceGeometryData3D, root_node: Node, callback: Callable = Callable()) 🔗

Parses the SceneTree for source geometry according to the properties of `navigation_mesh`. Updates the provided `source_geometry_data` resource with the resulting data. The resource can then be used to bake a navigation mesh with bake_from_source_geometry_data. After the process is finished the optional `callback` will be called.

Note: This function needs to run on the main thread or with a deferred call as the SceneTree is not thread-safe.

Performance: While convenient, reading data arrays from Mesh resources can affect the frame rate negatively. The data needs to be received from the GPU, stalling the RenderingServer in the process. For performance prefer the use of e.g. collision shapes or creating the data arrays entirely in code.

NavigationPathQueryParameters2D

Experimental: This class may be changed or removed in future versions.

Inherits: RefCounted **<** Object

Provides parameters for 2D navigation path queries.

Description

By changing various properties of this object, such as the start and target position, you can configure path queries to the NavigationServer2D.

Tutorials

- Using NavigationPathQueryObjects

Properties

RID	map	`RID()`
BitField[PathMetadataFlags]	metadata_flags	`7`
int	navigation_layers	`1`

PathPostProcessing	path_postprocessing	`0`
PathfindingAlgorithm	pathfinding_algorithm	`0`
float	simplify_epsilon	`0.0`
bool	simplify_path	`false`
Vector2	start_position	`Vector2(0, 0)`
Vector2	target_position	`Vector2(0, 0)`

Enumerations

enum **PathfindingAlgorithm**: 🔗

PathfindingAlgorithm **PATHFINDING_ALGORITHM_ASTAR** = `0`

The path query uses the default A* pathfinding algorithm.

---

enum **PathPostProcessing**: 🔗

PathPostProcessing
PATH_POSTPROCESSING_CORRIDORFUNNEL = `0`

Applies a funnel algorithm to the raw path corridor found by the pathfinding algorithm. This will result in the shortest path possible inside the path corridor. This postprocessing very much depends on the navigation mesh polygon layout and the created corridor. Especially tile- or gridbased layouts can face artificial corners with diagonal movement due to a jagged path corridor imposed by the cell shapes.

PathPostProcessing **PATH_POSTPROCESSING_EDGECENTERED** = `1`

Centers every path position in the middle of the traveled navigation mesh polygon edge. This creates better paths for tile- or gridbased layouts that restrict the movement to the cells center.

---

flags **PathMetadataFlags**: 🔗

PathMetadataFlags **PATH_METADATA_INCLUDE_NONE** = `0`

Don't include any additional metadata about the returned path.

PathMetadataFlags **PATH_METADATA_INCLUDE_TYPES** = `1`

Include the type of navigation primitive (region or link) that each point of the path goes through.

PathMetadataFlags **PATH_METADATA_INCLUDE_RIDS** = `2`

Include the RIDs of the regions and links that each point of the path goes through.

PathMetadataFlags **PATH_METADATA_INCLUDE_OWNERS** = `4`

Include the `ObjectID`s of the Objects which manage the regions and links each point of the path goes through.

PathMetadataFlags **PATH_METADATA_INCLUDE_ALL** = `7`

Include all available metadata about the returned path.

Property Descriptions

RID **map** = `RID()` 🔗

- void **set_map**(value: RID)
- RID **get_map**()

The navigation map RID used in the path query.

BitField[PathMetadataFlags] **metadata_flags** = `7` 🔗

- void **set_metadata_flags**(value: BitField[PathMetadataFlags])
- BitField[PathMetadataFlags] **get_metadata_flags**()

Additional information to include with the navigation path.

---

int **navigation_layers** = `1` 🔗

- void **set_navigation_layers**(value: int)
- int **get_navigation_layers**()

The navigation layers the query will use (as a bitmask).

PathPostProcessing **path_postprocessing** = `0` 🔗

- void **set_path_postprocessing**(value: PathPostProcessing)
- PathPostProcessing **get_path_postprocessing**()

The path postprocessing applied to the raw path corridor found by the pathfinding_algorithm.

PathfindingAlgorithm **pathfinding_algorithm** = `0`

- void **set_pathfinding_algorithm**(value: PathfindingAlgorithm)
- PathfindingAlgorithm **get_pathfinding_algorithm**()

The pathfinding algorithm used in the path query.

---

float **simplify_epsilon** = `0.0`

- void **set_simplify_epsilon**(value: float)
- float **get_simplify_epsilon**()

The path simplification amount in worlds units.

---

bool **simplify_path** = `false`

- void **set_simplify_path**(value: bool)
- bool **get_simplify_path**()

If `true` a simplified version of the path will be returned with less critical path points removed. The simplification amount is controlled by simplify_epsilon. The simplification uses a variant of Ramer-Douglas-Peucker algorithm for curve point decimation.

Path simplification can be helpful to mitigate various path following issues that can arise with certain agent types and script behaviors. E.g. "steering" agents or avoidance in "open fields".

Vector2 **start_position** = `Vector2(0, 0)`

- void **set_start_position**(value: Vector2)
- Vector2 **get_start_position**()

The pathfinding start position in global coordinates.

Vector2 **target_position** = `Vector2(0, 0)` 🔗

- void **set_target_position**(value: Vector2)
- Vector2 **get_target_position**()

The pathfinding target position in global coordinates.

NavigationPathQueryParameters3D

Experimental: This class may be changed or removed in future versions.

Inherits: RefCounted **<** Object

Provides parameters for 3D navigation path queries.

Description

By changing various properties of this object, such as the start and target position, you can configure path queries to the NavigationServer3D.

Tutorials

- Using NavigationPathQueryObjects

Properties

RID	map	`RID()`
BitField[PathMetadataFlags]	metadata_flags	`7`
int	navigation_layers	`1`

PathPostProcessing	path_postprocessing	`0`
PathfindingAlgorithm	pathfinding_algorithm	`0`
float	simplify_epsilon	`0.0`
bool	simplify_path	`false`
Vector3	start_position	`Vector3(0, 0, 0)`
Vector3	target_position	`Vector3(0, 0, 0)`

Enumerations

enum **PathfindingAlgorithm**: 🔗

PathfindingAlgorithm **PATHFINDING_ALGORITHM_ASTAR** = `0`

The path query uses the default A* pathfinding algorithm.

enum **PathPostProcessing**: 🔗

PathPostProcessing **PATH_POSTPROCESSING_CORRIDORFUNNEL** = `0`

Applies a funnel algorithm to the raw path corridor found by the pathfinding algorithm. This will result in the shortest path possible inside the path corridor. This postprocessing very much depends on the navigation mesh polygon layout and the created corridor. Especially tile- or gridbased layouts can face artificial corners with diagonal movement due to a jagged path corridor imposed by the cell shapes.

PathPostProcessing **PATH_POSTPROCESSING_EDGECENTERED** = `1`

Centers every path position in the middle of the traveled navigation mesh polygon edge. This creates better paths for tile- or gridbased layouts that restrict the movement to the cells center.

---

flags **PathMetadataFlags**: 🔗

PathMetadataFlags **PATH_METADATA_INCLUDE_NONE** = `0`

Don't include any additional metadata about the returned path.

PathMetadataFlags **PATH_METADATA_INCLUDE_TYPES** = `1`

Include the type of navigation primitive (region or link) that each point of the path goes through.

PathMetadataFlags **PATH_METADATA_INCLUDE_RIDS** = `2`

Include the RIDs of the regions and links that each point of the path goes through.

PathMetadataFlags **PATH_METADATA_INCLUDE_OWNERS** = `4`

Include the `ObjectID`s of the Objects which manage the regions and links each point of the path goes through.

PathMetadataFlags **PATH_METADATA_INCLUDE_ALL** = `7`

Include all available metadata about the returned path.

Property Descriptions

RID **map** = `RID()`

- void **set_map**(value: RID)
- RID **get_map**()

The navigation map RID used in the path query.

BitField[PathMetadataFlags] **metadata_flags** = `7`

- void **set_metadata_flags**(value: BitField[PathMetadataFlags])
- BitField[PathMetadataFlags] **get_metadata_flags**()

Additional information to include with the navigation path.

---

int **navigation_layers** = `1`

- void **set_navigation_layers**(value: int)
- int **get_navigation_layers**()

The navigation layers the query will use (as a bitmask).

PathPostProcessing **path_postprocessing** = `0`

- void **set_path_postprocessing**(value: PathPostProcessing)
- PathPostProcessing **get_path_postprocessing**()

The path postprocessing applied to the raw path corridor found by the pathfinding_algorithm.

PathfindingAlgorithm **pathfinding_algorithm** = `0`

- void **set_pathfinding_algorithm**(value: PathfindingAlgorithm)
- PathfindingAlgorithm **get_pathfinding_algorithm**()

The pathfinding algorithm used in the path query.

float **simplify_epsilon** = `0.0`

- void **set_simplify_epsilon**(value: float)
- float **get_simplify_epsilon**()

The path simplification amount in worlds units.

bool **simplify_path** = `false`

- void **set_simplify_path**(value: bool)
- bool **get_simplify_path**()

If `true` a simplified version of the path will be returned with less critical path points removed. The simplification amount is controlled by simplify_epsilon. The simplification uses a variant of Ramer-Douglas-Peucker algorithm for curve point decimation.

Path simplification can be helpful to mitigate various path following issues that can arise with certain agent types and script behaviors. E.g. "steering" agents or avoidance in "open fields".

Vector3 **start_position** = `Vector3(0, 0, 0)`

- void **set_start_position**(value: Vector3)
- Vector3 **get_start_position**()

The pathfinding start position in global coordinates.

Vector3 **target_position** = `Vector3(0, 0, 0)` 🔗

- void **set_target_position**(value: Vector3)
- Vector3 **get_target_position**()

The pathfinding target position in global coordinates.

NavigationPathQueryResult2D

Experimental: This class may be changed or removed in future versions.

Inherits: RefCounted **<** Object

Represents the result of a 2D pathfinding query.

Description

This class stores the result of a 2D navigation path query from the NavigationServer2D.

Tutorials

- Using NavigationPathQueryObjects

Properties

PackedVector2Array	path	`PackedVector2Array()`
PackedInt64Array	path_owner_ids	`PackedInt64Array()`
Array[RID]	path_rids	`[]`

PackedInt32Array	path_types	`PackedInt32Array()`

Methods

void	reset()

Enumerations

enum **PathSegmentType**: 🔗

PathSegmentType **PATH_SEGMENT_TYPE_REGION** = `0`

This segment of the path goes through a region.

PathSegmentType **PATH_SEGMENT_TYPE_LINK** = `1`

This segment of the path goes through a link.

Property Descriptions

PackedVector2Array **path** = `PackedVector2Array()` 🔗

- void **set_path**(value: PackedVector2Array)
- PackedVector2Array **get_path**()

The resulting path array from the navigation query. All path array positions are in global coordinates. Without customized query parameters this is the same path as returned by NavigationServer2D.map_get_path.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedVector2Array for more details.

PackedInt64Array **path_owner_ids** = `PackedInt64Array()`

- void **set_path_owner_ids**(value: PackedInt64Array)
- PackedInt64Array **get_path_owner_ids**()

The `ObjectID`s of the Objects which manage the regions and links each point of the path goes through.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedInt64Array for more details.

Array[RID] **path_rids** = `[]`

- void **set_path_rids**(value: Array[RID])
- Array[RID] **get_path_rids**()

The RIDs of the regions and links that each point of the path goes through.

PackedInt32Array **path_types** = `PackedInt32Array()`

- void **set_path_types**(value: PackedInt32Array)
- PackedInt32Array **get_path_types**()

The type of navigation primitive (region or link) that each point of the path goes through.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedInt32Array for more details.

Method Descriptions

void **reset**() 🔗

Reset the result object to its initial state. This is useful to reuse the object across multiple queries.

NavigationPathQueryResult3D

Experimental: This class may be changed or removed in future versions.

Inherits: RefCounted **<** Object

Represents the result of a 3D pathfinding query.

Description

This class stores the result of a 3D navigation path query from the NavigationServer3D.

Tutorials

- Using NavigationPathQueryObjects

Properties

PackedVector3Array	path	`PackedVector3Array()`
PackedInt64Array	path_owner_ids	`PackedInt64Array()`
Array[RID]	path_rids	`[]`

PackedInt32Array	path_types	`PackedInt32Array()`

Methods

void	reset()

Enumerations

enum **PathSegmentType**: 🔗

PathSegmentType **PATH_SEGMENT_TYPE_REGION** = `0`

This segment of the path goes through a region.

PathSegmentType **PATH_SEGMENT_TYPE_LINK** = `1`

This segment of the path goes through a link.

Property Descriptions

PackedVector3Array **path** = `PackedVector3Array()` 🔗

- void **set_path**(value: PackedVector3Array)
- PackedVector3Array **get_path**()

The resulting path array from the navigation query. All path array positions are in global coordinates. Without customized query parameters this is the same path as returned by NavigationServer3D.map_get_path.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedVector3Array for more details.

PackedInt64Array **path_owner_ids** = `PackedInt64Array()`

- void **set_path_owner_ids**(value: PackedInt64Array)
- PackedInt64Array **get_path_owner_ids**()

The `ObjectID`s of the Objects which manage the regions and links each point of the path goes through.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedInt64Array for more details.

Array[RID] **path_rids** = `[]`

- void **set_path_rids**(value: Array[RID])
- Array[RID] **get_path_rids**()

The RIDs of the regions and links that each point of the path goes through.

PackedInt32Array **path_types** = `PackedInt32Array()`

- void **set_path_types**(value: PackedInt32Array)
- PackedInt32Array **get_path_types**()

The type of navigation primitive (region or link) that each point of the path goes through.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedInt32Array for more details.

Method Descriptions

void **reset**() 🔗

Reset the result object to its initial state. This is useful to reuse the object across multiple queries.

NavigationServer2D

Experimental: This class may be changed or removed in future versions.

Inherits: Object

A server interface for low-level 2D navigation access.

Description

NavigationServer2D is the server that handles navigation maps, regions and agents. It does not handle A* navigation from AStar2D or AStarGrid2D.

Maps are divided into regions, which are composed of navigation polygons. Together, they define the traversable areas in the 2D world.

Note: Most **NavigationServer2D** changes take effect after the next physics frame and not immediately. This includes all changes made to maps, regions or agents by navigation-related nodes in the scene tree or made through scripts.

For two regions to be connected to each other, they must share a similar edge. An edge is considered connected to another if both of its two vertices are at a distance less than `edge_connection_margin` to the respective other edge's vertex.

You may assign navigation layers to regions with region_set_navigation_layers, which then can be checked upon when requesting a path with map_get_path. This can be used to allow or deny certain areas for some objects.

To use the collision avoidance system, you may use agents. You can set an agent's target velocity, then the servers will emit a callback with a modified velocity.

Note: The collision avoidance system ignores regions. Using the modified velocity directly may move an agent outside of the traversable area. This is a limitation of the collision avoidance system, any more complex situation may require the use of the physics engine.

This server keeps tracks of any call and executes them during the sync phase. This means that you can request any change to the map, using any thread, without worrying.

Tutorials

- Using NavigationServer
- Navigation Polygon 2D Demo [https://godotengine.org/asset-library/asset/2722]

Methods

RID	agent_create()
bool	agent_get_avoidance_enabled(agent: RID) const
int	agent_get_avoidance_layers(agent: RID) const
int	agent_get_avoidance_mask(agent: RID) const
float	agent_get_avoidance_priority(agent: RID) const
RID	agent_get_map(agent: RID) const
int	agent_get_max_neighbors(agent: RID) const
float	agent_get_max_speed(agent: RID) const
float	agent_get_neighbor_distance(agent: RID) const
bool	agent_get_paused(agent: RID) const
Vector2	agent_get_position(agent: RID) const
float	agent_get_radius(agent: RID) const
float	agent_get_time_horizon_agents(agent: RID) const

float	agent_get_time_horizon_obstacles(agent: RID) const
Vector2	agent_get_velocity(agent: RID) const
bool	agent_has_avoidance_callback(agent: RID) const
bool	agent_is_map_changed(agent: RID) const
void	agent_set_avoidance_callback(agent: RID, callback: Callable)
void	agent_set_avoidance_enabled(agent: RID, enabled: bool)
void	agent_set_avoidance_layers(agent: RID, layers: int)
void	agent_set_avoidance_mask(agent: RID, mask: int)
void	agent_set_avoidance_priority(agent: RID, priority: float)
void	agent_set_map(agent: RID, map: RID)
void	agent_set_max_neighbors(agent: RID, count: int)
void	agent_set_max_speed(agent: RID, max_speed: float)
void	agent_set_neighbor_distance(agent: RID, distance: float)
void	agent_set_paused(agent: RID, paused: bool)
void	agent_set_position(agent: RID, position: Vector2)
void	agent_set_radius(agent: RID, radius: float)

void	agent_set_time_horizon_agents(agent: RID, time_horizon: float)
void	agent_set_time_horizon_obstacles(agent: RID, time_horizon: float)
void	agent_set_velocity(agent: RID, velocity: Vector2)
void	agent_set_velocity_forced(agent: RID, velocity: Vector2)
void	bake_from_source_geometry_data(navigation_polygon: NavigationPolygon, source_geometry_data: NavigationMeshSourceGeometryData2D, callback: Callable = Callable())
void	bake_from_source_geometry_data_async(navigation_polygon: NavigationPolygon, source_geometry_data: NavigationMeshSourceGeometryData2D, callback: Callable = Callable())
void	free_rid(rid: RID)
bool	get_debug_enabled() const
Array[RID]	get_maps() const
bool	is_baking_navigation_polygon(navigation_polygon: NavigationPolygon) const
RID	link_create()
bool	link_get_enabled(link: RID) const

Vector2	link_get_end_position(link: RID) const
float	link_get_enter_cost(link: RID) const
RID	link_get_map(link: RID) const
int	link_get_navigation_layers(link: RID) const
int	link_get_owner_id(link: RID) const
Vector2	link_get_start_position(link: RID) const
float	link_get_travel_cost(link: RID) const
bool	link_is_bidirectional(link: RID) const
void	link_set_bidirectional(link: RID, bidirectional: bool)
void	link_set_enabled(link: RID, enabled: bool)
void	link_set_end_position(link: RID, position: Vector2)
void	link_set_enter_cost(link: RID, enter_cost: float)
void	link_set_map(link: RID, map: RID)
void	link_set_navigation_layers(link: RID, navigation_layers: int)
void	link_set_owner_id(link: RID, owner_id: int)
void	link_set_start_position(link: RID, position: Vector2)

void	link_set_travel_cost(link: RID, travel_cost: float)
RID	map_create()
void	map_force_update(map: RID)
Array[RID]	map_get_agents(map: RID) const
float	map_get_cell_size(map: RID) const
Vector2	map_get_closest_point(map: RID, to_point: Vector2) const
RID	map_get_closest_point_owner(map: RID, to_point: Vector2) const
float	map_get_edge_connection_margin(map: RID) const
int	map_get_iteration_id(map: RID) const
float	map_get_link_connection_radius(map: RID) const
Array[RID]	map_get_links(map: RID) const
Array[RID]	map_get_obstacles(map: RID) const
PackedVector2Array	map_get_path(map: RID, origin: Vector2, destination: Vector2, optimize: bool, navigation_layers: int = 1) const
Vector2	map_get_random_point(map: RID, navigation_layers: int, uniformly: bool) const

Array[RID]	map_get_regions(map: RID) const
bool	map_get_use_edge_connections(map: RID) const
bool	map_is_active(map: RID) const
void	map_set_active(map: RID, active: bool)
void	map_set_cell_size(map: RID, cell_size: float)
void	map_set_edge_connection_margin(map: RID, margin: float)
void	map_set_link_connection_radius(map: RID, radius: float)
void	map_set_use_edge_connections(map: RID, enabled: bool)
RID	obstacle_create()
bool	obstacle_get_avoidance_enabled(obstacle: RID) const
int	obstacle_get_avoidance_layers(obstacle: RID) const
RID	obstacle_get_map(obstacle: RID) const
bool	obstacle_get_paused(obstacle: RID) const
Vector2	obstacle_get_position(obstacle: RID) const
float	obstacle_get_radius(obstacle: RID) const
Vector2	obstacle_get_velocity(obstacle: RID) const

PackedVector2Array	obstacle_get_vertices(obstacle: RID) const
void	obstacle_set_avoidance_enabled(obstacle: RID, enabled: bool)
void	obstacle_set_avoidance_layers(obstacle: RID, layers: int)
void	obstacle_set_map(obstacle: RID, map: RID)
void	obstacle_set_paused(obstacle: RID, paused: bool)
void	obstacle_set_position(obstacle: RID, position: Vector2)
void	obstacle_set_radius(obstacle: RID, radius: float)
void	obstacle_set_velocity(obstacle: RID, velocity: Vector2)
void	obstacle_set_vertices(obstacle: RID, vertices: PackedVector2Array)
void	parse_source_geometry_data(navigation_polygon: NavigationPolygon, source_geometry_data: NavigationMeshSourceGeometryData2D, root_node: Node, callback: Callable = Callable())
void	query_path(parameters: NavigationPathQueryParameters2D, result: NavigationPathQueryResult2D) const
RID	region_create()
Vector2	region_get_connection_pathway_end(region: RID, connection: int) const

Vector2	region_get_connection_pathway_start(region: RID, connection: int) const
int	region_get_connections_count(region: RID) const
bool	region_get_enabled(region: RID) const
float	region_get_enter_cost(region: RID) const
RID	region_get_map(region: RID) const
int	region_get_navigation_layers(region: RID) const
int	region_get_owner_id(region: RID) const
Vector2	region_get_random_point(region: RID, navigation_layers: int, uniformly: bool) const
Transform2D	region_get_transform(region: RID) const
float	region_get_travel_cost(region: RID) const
bool	region_get_use_edge_connections(region: RID) const
bool	region_owns_point(region: RID, point: Vector2) const
void	region_set_enabled(region: RID, enabled: bool)
void	region_set_enter_cost(region: RID, enter_cost: float)
void	region_set_map(region: RID, map: RID)

void	region_set_navigation_layers(region: RID, navigation_layers: int)
void	region_set_navigation_polygon(region: RID, navigation_polygon: NavigationPolygon)
void	region_set_owner_id(region: RID, owner_id: int)
void	region_set_transform(region: RID, transform: Transform2D)
void	region_set_travel_cost(region: RID, travel_cost: float)
void	region_set_use_edge_connections(region: RID, enabled: bool)
void	set_debug_enabled(enabled: bool)
PackedVector2Array	simplify_path(path: PackedVector2Array, epsilon: float)
RID	source_geometry_parser_create()
void	source_geometry_parser_set_callback(parser: RID, callback: Callable)

Signals

map_changed(map: RID) 🔗

Emitted when a navigation map is updated, when a region moves or is modified.

navigation_debug_changed() 🔗

Emitted when navigation debug settings are changed. Only available in debug builds.

Method Descriptions

RID **agent_create**()

Creates the agent.

---

bool **agent_get_avoidance_enabled**(agent: RID) const

Return `true` if the specified `agent` uses avoidance.

---

int **agent_get_avoidance_layers**(agent: RID) const

Returns the `avoidance_layers` bitmask of the specified `agent`.

---

int **agent_get_avoidance_mask**(agent: RID) const

Returns the `avoidance_mask` bitmask of the specified `agent`.

---

float **agent_get_avoidance_priority**(agent: RID) const

Returns the `avoidance_priority` of the specified `agent`.

RID **agent_get_map**(agent: RID) const

Returns the navigation map RID the requested `agent` is currently assigned to.

---

int **agent_get_max_neighbors**(agent: RID) const

Returns the maximum number of other agents the specified `agent` takes into account in the navigation.

---

float **agent_get_max_speed**(agent: RID) const

Returns the maximum speed of the specified `agent`.

---

float **agent_get_neighbor_distance**(agent: RID) const

Returns the maximum distance to other agents the specified `agent` takes into account in the navigation.

---

bool **agent_get_paused**(agent: RID) const

Returns `true` if the specified `agent` is paused.

Vector2 **agent_get_position**(agent: RID) const

Returns the position of the specified `agent` in world space.

---

float **agent_get_radius**(agent: RID) const

Returns the radius of the specified `agent`.

---

float **agent_get_time_horizon_agents**(agent: RID) const

Returns the minimal amount of time for which the specified `agent`'s velocities that are computed by the simulation are safe with respect to other agents.

---

float **agent_get_time_horizon_obstacles**(agent: RID) const

Returns the minimal amount of time for which the specified `agent`'s velocities that are computed by the simulation are safe with respect to static avoidance obstacles.

Vector2 **agent_get_velocity**(agent: RID) const

Returns the velocity of the specified `agent`.

---

bool **agent_has_avoidance_callback**(agent: RID) const

Return `true` if the specified `agent` has an avoidance callback.

---

bool **agent_is_map_changed**(agent: RID) const 🔗

Returns true if the map got changed the previous frame.

---

void **agent_set_avoidance_callback**(agent: RID, callback: Callable) 🔗

Sets the callback Callable that gets called after each avoidance processing step for the `agent`. The calculated `safe_velocity` will be dispatched with a signal to the object just before the physics calculations.

Note: Created callbacks are always processed independently of the SceneTree state as long as the agent is on a navigation map and not freed. To disable the dispatch of a callback from an agent use agent_set_avoidance_callback again with an empty Callable.

---

void **agent_set_avoidance_enabled**(agent: RID, enabled: bool) 🔗

If `enabled` is `true`, the specified `agent` uses avoidance.

---

void **agent_set_avoidance_layers**(agent: RID, layers: int) 🔗

Set the agent's `avoidance_layers` bitmask.

---

void **agent_set_avoidance_mask**(agent: RID, mask: int) 🔗

Set the agent's `avoidance_mask` bitmask.

---

void **agent_set_avoidance_priority**(agent: RID, priority: float) 🔗

Set the agent's `avoidance_priority` with a `priority` between 0.0 (lowest priority) to 1.0 (highest priority).

The specified `agent` does not adjust the velocity for other agents that would match the `avoidance_mask` but have a lower `avoidance_priority`. This in turn makes the other agents with lower priority adjust their velocities even more to avoid collision with this agent.

---

void **agent_set_map**(agent: RID, map: RID) 🔗

Puts the agent in the map.

---

void **agent_set_max_neighbors**(agent: RID, count: int)

Sets the maximum number of other agents the agent takes into account in the navigation. The larger this number, the longer the running time of the simulation. If the number is too low, the simulation will not be safe.

---

void **agent_set_max_speed**(agent: RID, max_speed: float)

Sets the maximum speed of the agent. Must be positive.

---

void **agent_set_neighbor_distance**(agent: RID, distance: float)

Sets the maximum distance to other agents this agent takes into account in the navigation. The larger this number, the longer the running time of the simulation. If the number is too low, the simulation will not be safe.

---

void **agent_set_paused**(agent: RID, paused: bool)

If `paused` is true the specified `agent` will not be processed, e.g. calculate avoidance velocities or receive avoidance callbacks.

---

void **agent_set_position**(agent: RID, position: Vector2)

Sets the position of the agent in world space.

---

void **agent_set_radius**(agent: RID, radius: float)

Sets the radius of the agent.

---

void **agent_set_time_horizon_agents**(agent: RID, time_horizon: float)

The minimal amount of time for which the agent's velocities that are computed by the simulation are safe with respect to other agents. The larger this number, the sooner this agent will respond to the presence of other agents, but the less freedom this agent has in choosing its velocities. A too high value will slow down agents movement considerably. Must be positive.

---

void **agent_set_time_horizon_obstacles**(agent: RID, time_horizon: float)

The minimal amount of time for which the agent's velocities that are computed by the simulation are safe with respect to static avoidance obstacles. The larger this number, the sooner this agent will respond to the presence of static avoidance obstacles, but the less freedom this agent has in choosing its velocities. A too high value will slow down agents movement considerably. Must be positive.

---

void **agent_set_velocity**(agent: RID, velocity: Vector2)

Sets `velocity` as the new wanted velocity for the specified `agent`. The avoidance simulation will try to fulfill this velocity if possible but will modify it to avoid collision with other agent's and obstacles. When an agent is teleported to a new position far away use agent_set_velocity_forced instead to reset the internal velocity state.

---

void **agent_set_velocity_forced**(agent: RID, velocity: Vector2)

Replaces the internal velocity in the collision avoidance simulation with `velocity` for the specified `agent`. When an agent is teleported to a new position far away this function should be used in the same frame. If called frequently this function can get agents stuck.

---

void **bake_from_source_geometry_data**(navigation_polygon: NavigationPolygon, source_geometry_data: NavigationMeshSourceGeometryData2D, callback: Callable = Callable())

Bakes the provided `navigation_polygon` with the data from the provided `source_geometry_data`. After the process is finished the optional `callback` will be called.

---

void **bake_from_source_geometry_data_async**(navigation_polygon: NavigationPolygon, source_geometry_data: NavigationMeshSourceGeometryData2D, callback: Callable = Callable())

Bakes the provided `navigation_polygon` with the data from the provided `source_geometry_data` as an async task running on a background thread. After the process is finished the optional `callback` will be called.

---

void **free_rid**(rid: RID)

Destroys the given RID.

---

bool **get_debug_enabled**() const

Returns `true` when the NavigationServer has debug enabled.

Array[RID] **get_maps**() const

Returns all created navigation map RIDs on the NavigationServer. This returns both 2D and 3D created navigation maps as there is technically no distinction between them.

---

bool **is_baking_navigation_polygon**(navigation_polygon: NavigationPolygon) const

Returns `true` when the provided navigation polygon is being baked on a background thread.

RID **link_create**()

Create a new link between two positions on a map.

---

bool **link_get_enabled**(link: RID) const

Returns `true` if the specified `link` is enabled.

Vector2 **link_get_end_position**(link: RID) const

Returns the ending position of this `link`.

---

float **link_get_enter_cost**(link: RID) const

Returns the enter cost of this `link`.

RID **link_get_map**(link: RID) const

Returns the navigation map RID the requested `link` is currently assigned to.

int **link_get_navigation_layers**(link: RID) const

Returns the navigation layers for this `link`.

int **link_get_owner_id**(link: RID) const

Returns the `ObjectID` of the object which manages this link.

Vector2 **link_get_start_position**(link: RID) const

Returns the starting position of this `link`.

float **link_get_travel_cost**(link: RID) const

Returns the travel cost of this `link`.

bool **link_is_bidirectional**(link: RID) const

Returns whether this `link` can be travelled in both directions.

void **link_set_bidirectional**(link: RID, bidirectional: bool)

Sets whether this `link` can be travelled in both directions.

void **link_set_enabled**(link: RID, enabled: bool)

If `enabled` is `true`, the specified `link` will contribute to its current navigation map.

void **link_set_end_position**(link: RID, position: Vector2)

Sets the exit position for the `link`.

void **link_set_enter_cost**(link: RID, enter_cost: float)

Sets the `enter_cost` for this `link`.

void **link_set_map**(link: RID, map: RID)

Sets the navigation map RID for the link.

void **link_set_navigation_layers**(link: RID, navigation_layers: int)

Set the links's navigation layers. This allows selecting links from a path request (when using map_get_path).

void **link_set_owner_id**(link: RID, owner_id: int)

Set the `ObjectID` of the object which manages this link.

void **link_set_start_position**(link: RID, position: Vector2)

Sets the entry position for this `link`.

void **link_set_travel_cost**(link: RID, travel_cost: float)

Sets the `travel_cost` for this `link`.

RID **map_create**()

Create a new map.

void **map_force_update**(map: RID)

This function immediately forces synchronization of the specified navigation `map` RID. By default navigation maps are only synchronized at the end of each physics frame. This function can be used to immediately (re)calculate all the navigation meshes and region connections of the navigation map. This makes it possible to query a navigation path for a changed map immediately and in the same frame (multiple times if needed).

Due to technical restrictions the current NavigationServer command queue will be flushed. This means all already queued update commands for this physics frame will be executed, even those intended for other maps, regions and agents not part of the specified map. The expensive computation of the navigation meshes and region connections of a map will only be done for the specified map. Other maps will receive the normal synchronization at the end of the physics frame. Should the specified map receive changes after the forced update it will update again as well when the other maps receive their update.

Avoidance processing and dispatch of the `safe_velocity` signals is unaffected by this function and continues to happen for all maps and agents at the end of the physics frame.

Note: With great power comes great responsibility. This function should only be used by users that really know what they are doing and have a good reason for it. Forcing an immediate update of a navigation map requires locking the NavigationServer and flushing the entire NavigationServer command queue. Not only can this severely impact the performance of a game but it can also introduce bugs if used inappropriately without much foresight.

Array[RID] **map_get_agents**(map: RID) const 🔗

Returns all navigation agents RIDs that are currently assigned to the requested navigation `map`.

---

float **map_get_cell_size**(map: RID) const 🔗

Returns the map cell size used to rasterize the navigation mesh vertices.

Vector2 **map_get_closest_point**(map: RID, to_point: Vector2) const 🔗

Returns the point closest to the provided `to_point` on the navigation mesh surface.

RID **map_get_closest_point_owner**(map: RID, to_point: Vector2) const 🔗

Returns the owner region RID for the point returned by map_get_closest_point.

---

float **map_get_edge_connection_margin**(map: RID) const 🔗

Returns the edge connection margin of the map. The edge connection margin is a distance used to connect two regions.

---

int **map_get_iteration_id**(map: RID) const

Returns the current iteration id of the navigation map. Every time the navigation map changes and synchronizes the iteration id increases. An iteration id of 0 means the navigation map has never synchronized.

Note: The iteration id will wrap back to 1 after reaching its range limit.

---

float **map_get_link_connection_radius**(map: RID) const

Returns the link connection radius of the map. This distance is the maximum range any link will search for navigation mesh polygons to connect to.

Array[RID] **map_get_links**(map: RID) const

Returns all navigation link RIDs that are currently assigned to the requested navigation `map`.

Array[RID] **map_get_obstacles**(map: RID) const

Returns all navigation obstacle RIDs that are currently assigned to the requested navigation `map`.

PackedVector2Array **map_get_path**(map: RID, origin: Vector2, destination: Vector2, optimize: bool, navigation_layers: int = 1) const

Returns the navigation path to reach the destination from the origin. `navigation_layers` is a bitmask of all region navigation layers that are allowed to be in the path.

Vector2 **map_get_random_point**(map: RID, navigation_layers: int, uniformly: bool) const

Returns a random position picked from all map region polygons with matching `navigation_layers`.

If `uniformly` is `true`, all map regions, polygons, and faces are weighted by their surface area (slower).

If `uniformly` is `false`, just a random region and a random polygon are picked (faster).

Array[RID] **map_get_regions**(map: RID) const

Returns all navigation regions RIDs that are currently assigned to the requested navigation `map`.

---

bool **map_get_use_edge_connections**(map: RID) const

Returns whether the navigation `map` allows navigation regions to use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin.

---

bool **map_is_active**(map: RID) const

Returns true if the map is active.

---

void **map_set_active**(map: RID, active: bool)

Sets the map active.

---

void **map_set_cell_size**(map: RID, cell_size: float)

Sets the map cell size used to rasterize the navigation mesh vertices. Must match with the cell size of the used navigation meshes.

---

void **map_set_edge_connection_margin**(map: RID, margin: float)

Set the map edge connection margin used to weld the compatible region edges.

---

void **map_set_link_connection_radius**(map: RID, radius: float)

Set the map's link connection radius used to connect links to navigation polygons.

---

void **map_set_use_edge_connections**(map: RID, enabled: bool)

Set the navigation `map` edge connection use. If `enabled` is `true`, the navigation map allows navigation regions to use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin.

RID **obstacle_create**()

Creates a new navigation obstacle.

bool **obstacle_get_avoidance_enabled**(obstacle: RID) const

Returns `true` if the provided `obstacle` has avoidance enabled.

int **obstacle_get_avoidance_layers**(obstacle: RID) const

Returns the `avoidance_layers` bitmask of the specified `obstacle`.

RID **obstacle_get_map**(obstacle: RID) const

Returns the navigation map RID the requested `obstacle` is currently assigned to.

bool **obstacle_get_paused**(obstacle: RID) const

Returns `true` if the specified `obstacle` is paused.

Vector2 **obstacle_get_position**(obstacle: RID) const

Returns the position of the specified `obstacle` in world space.

float **obstacle_get_radius**(obstacle: RID) const

Returns the radius of the specified dynamic `obstacle`.

Vector2 **obstacle_get_velocity**(obstacle: RID) const

Returns the velocity of the specified dynamic `obstacle`.

PackedVector2Array **obstacle_get_vertices**(obstacle: RID) const

Returns the outline vertices for the specified `obstacle`.

---

void **obstacle_set_avoidance_enabled**(obstacle: RID, enabled: bool)

If `enabled` is `true`, the provided `obstacle` affects avoidance using agents.

---

void **obstacle_set_avoidance_layers**(obstacle: RID, layers: int)

Set the obstacles's `avoidance_layers` bitmask.

---

void **obstacle_set_map**(obstacle: RID, map: RID)

Sets the navigation map RID for the obstacle.

---

void **obstacle_set_paused**(obstacle: RID, paused: bool)

If `paused` is true the specified `obstacle` will not be processed, e.g. affect avoidance velocities.

---

void **obstacle_set_position**(obstacle: RID, position: Vector2)

Sets the position of the obstacle in world space.

---

void **obstacle_set_radius**(obstacle: RID, radius: float)

Sets the radius of the dynamic obstacle.

---

void **obstacle_set_velocity**(obstacle: RID, velocity: Vector2)

Sets `velocity` of the dynamic `obstacle`. Allows other agents to better predict the movement of the dynamic obstacle. Only works in combination with the radius of the obstacle.

---

void **obstacle_set_vertices**(obstacle: RID, vertices: PackedVector2Array)

Sets the outline vertices for the obstacle. If the vertices are winded in clockwise order agents will be pushed in by the obstacle, else they will be pushed out.

---

void **parse_source_geometry_data**(navigation_polygon: NavigationPolygon, source_geometry_data: NavigationMeshSourceGeometryData2D, root_node: Node, callback: Callable = Callable()) 🔗

Parses the SceneTree for source geometry according to the properties of `navigation_polygon`. Updates the provided `source_geometry_data` resource with the resulting data. The resource can then be used to bake a navigation mesh with bake_from_source_geometry_data. After the process is finished the optional `callback` will be called.

Note: This function needs to run on the main thread or with a deferred call as the SceneTree is not thread-safe.

Performance: While convenient, reading data arrays from Mesh resources can affect the frame rate negatively. The data needs to be received from the GPU, stalling the RenderingServer in the process. For performance prefer the use of e.g. collision shapes or creating the data arrays entirely in code.

---

void **query_path**(parameters: NavigationPathQueryParameters2D, result: NavigationPathQueryResult2D) const 🔗

Queries a path in a given navigation map. Start and target position and other parameters are defined through NavigationPathQueryParameters2D. Updates the provided NavigationPathQueryResult2D result object with the path among other results requested by the query.

RID **region_create**() 🔗

Creates a new region.

Vector2 **region_get_connection_pathway_end**(region: RID, connection: int) const 🔗

Returns the ending point of a connection door. `connection` is an index between 0 and the return value of region_get_connections_count.

Vector2 **region_get_connection_pathway_start**(region: RID, connection: int) const

Returns the starting point of a connection door. `connection` is an index between 0 and the return value of region_get_connections_count.

---

int **region_get_connections_count**(region: RID) const

Returns how many connections this `region` has with other regions in the map.

---

bool **region_get_enabled**(region: RID) const

Returns `true` if the specified `region` is enabled.

---

float **region_get_enter_cost**(region: RID) const

Returns the enter cost of this `region`.

RID **region_get_map**(region: RID) const

Returns the navigation map RID the requested `region` is currently assigned to.

---

int **region_get_navigation_layers**(region: RID) const

Returns the region's navigation layers.

---

int **region_get_owner_id**(region: RID) const

Returns the `ObjectID` of the object which manages this region.

Vector2 **region_get_random_point**(region: RID, navigation_layers: int, uniformly: bool) const

Returns a random position picked from all region polygons with matching `navigation_layers`.

If `uniformly` is `true`, all region polygons and faces are weighted by their surface area (slower).

If `uniformly` is `false`, just a random polygon and face is picked (faster).

Transform2D **region_get_transform**(region: RID) const

Returns the global transformation of this `region`.

---

float **region_get_travel_cost**(region: RID) const

Returns the travel cost of this `region`.

---

bool **region_get_use_edge_connections**(region: RID) const

Returns whether the navigation `region` is set to use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin.

---

bool **region_owns_point**(region: RID, point: Vector2) const

Returns `true` if the provided `point` in world space is currently owned by the provided navigation `region`. Owned in this context means that one of the region's navigation mesh polygon faces has a possible position at the closest distance to this point compared to all other navigation meshes from other navigation regions that are also registered on the navigation map of the provided region.

If multiple navigation meshes have positions at equal distance the navigation region whose polygons are processed first wins the ownership. Polygons are processed in the same order that navigation regions were registered on the NavigationServer.

Note: If navigation meshes from different navigation regions overlap (which should be avoided in general) the result might not be what is expected.

---

void **region_set_enabled**(region: RID, enabled: bool)

If `enabled` is `true` the specified `region` will contribute to its current navigation map.

---

void **region_set_enter_cost**(region: RID, enter_cost: float)

Sets the `enter_cost` for this `region`.

---

void **region_set_map**(region: RID, map: RID) 🔗

Sets the map for the region.

void **region_set_navigation_layers**(region: RID, navigation_layers: int) 🔗

Set the region's navigation layers. This allows selecting regions from a path request (when using map_get_path).

void **region_set_navigation_polygon**(region: RID, navigation_polygon: NavigationPolygon) 🔗

Sets the `navigation_polygon` for the region.

void **region_set_owner_id**(region: RID, owner_id: int) 🔗

Set the `ObjectID` of the object which manages this region.

void **region_set_transform**(region: RID, transform: Transform2D) 🔗

Sets the global transformation for the region.

void **region_set_travel_cost**(region: RID, travel_cost: float) 🔗

Sets the `travel_cost` for this `region`.

void **region_set_use_edge_connections**(region: RID, enabled: bool) 🔗

If `enabled` is `true`, the navigation `region` will use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin.

void **set_debug_enabled**(enabled: bool) 🔗

If `true` enables debug mode on the NavigationServer.

PackedVector2Array **simplify_path**(path: PackedVector2Array, epsilon: float) 🔗

Returns a simplified version of `path` with less critical path points removed. The simplification amount is in worlds units and controlled by `epsilon`. The simplification uses a variant of Ramer-Douglas-Peucker algorithm for curve point decimation.

Path simplification can be helpful to mitigate various path following issues that can arise with certain agent types and script behaviors. E.g. "steering" agents or avoidance in "open fields".

RID **source_geometry_parser_create**()

Creates a new source geometry parser. If a Callable is set for the parser with source_geometry_parser_set_callback the callback will be called for every single node that gets parsed whenever parse_source_geometry_data is used.

---

void **source_geometry_parser_set_callback**(parser: RID, callback: Callable)

Sets the `callback` Callable for the specific source geometry `parser`. The Callable will receive a call with the following parameters:

- `navigation_mesh` - The NavigationPolygon reference used to define the parse settings. Do NOT edit or add directly to the navigation mesh.
- `source_geometry_data` - The NavigationMeshSourceGeometryData2D reference. Add custom source geometry for navigation mesh baking to this object.
- `node` - The Node that is parsed.

NavigationServer3D

Experimental: This class may be changed or removed in future versions.

Inherits: Object

A server interface for low-level 3D navigation access.

Description

NavigationServer3D is the server that handles navigation maps, regions and agents. It does not handle A* navigation from AStar3D.

Maps are divided into regions, which are composed of navigation meshes. Together, they define the navigable areas in the 3D world.

Note: Most **NavigationServer3D** changes take effect after the next physics frame and not immediately. This includes all changes made to maps, regions or agents by navigation-related nodes in the scene tree or made through scripts.

For two regions to be connected to each other, they must share a similar edge. An edge is considered connected to another if both of its two vertices are at a distance less than `edge_connection_margin` to the respective other edge's vertex.

You may assign navigation layers to regions with region_set_navigation_layers, which then can be checked upon when requesting a path with map_get_path. This can be used to allow or deny certain areas for some objects.

To use the collision avoidance system, you may use agents. You can set an agent's target velocity, then the servers will emit a callback with a modified velocity.

Note: The collision avoidance system ignores regions. Using the modified velocity directly may move an agent outside of the traversable area. This is a limitation of the collision avoidance system, any more complex situation may require the use of the physics engine.

This server keeps tracks of any call and executes them during the sync phase. This means that you can request any change to the map, using any thread, without worrying.

Tutorials

- Using NavigationServer
- 3D Navigation Demo [https://godotengine.org/asset-library/asset/2743]

Methods

RID	agent_create()
bool	agent_get_avoidance_enabled(agent: RID) const
int	agent_get_avoidance_layers(agent: RID) const
int	agent_get_avoidance_mask(agent: RID) const
float	agent_get_avoidance_priority(agent: RID) const
float	agent_get_height(agent: RID) const
RID	agent_get_map(agent: RID) const
int	agent_get_max_neighbors(agent: RID) const
float	agent_get_max_speed(agent: RID) const
float	agent_get_neighbor_distance(agent: RID) const
bool	agent_get_paused(agent: RID) const
Vector3	agent_get_position(agent: RID) const

float	agent_get_radius(agent: RID) const
float	agent_get_time_horizon_agents(agent: RID) const
float	agent_get_time_horizon_obstacles(agent: RID) const
bool	agent_get_use_3d_avoidance(agent: RID) const
Vector3	agent_get_velocity(agent: RID) const
bool	agent_has_avoidance_callback(agent: RID) const
bool	agent_is_map_changed(agent: RID) const
void	agent_set_avoidance_callback(agent: RID, callback: Callable)
void	agent_set_avoidance_enabled(agent: RID, enabled: bool)
void	agent_set_avoidance_layers(agent: RID, layers: int)
void	agent_set_avoidance_mask(agent: RID, mask: int)
void	agent_set_avoidance_priority(agent: RID, priority: float)
void	agent_set_height(agent: RID, height: float)
void	agent_set_map(agent: RID, map: RID)
void	agent_set_max_neighbors(agent: RID, count: int)

void	agent_set_max_speed(agent: RID, max_speed: float)
void	agent_set_neighbor_distance(agent: RID, distance: float)
void	agent_set_paused(agent: RID, paused: bool)
void	agent_set_position(agent: RID, position: Vector3)
void	agent_set_radius(agent: RID, radius: float)
void	agent_set_time_horizon_agents(agent: RID, time_horizon: float)
void	agent_set_time_horizon_obstacles(agent: RID, time_horizon: float)
void	agent_set_use_3d_avoidance(agent: RID, enabled: bool)
void	agent_set_velocity(agent: RID, velocity: Vector3)
void	agent_set_velocity_forced(agent: RID, velocity: Vector3)
void	bake_from_source_geometry_data(navigation_mesh: NavigationMesh, source_geometry_data: NavigationMeshSourceGeometryData3D, callback: Callable = Callable())
void	bake_from_source_geometry_data_async(navigation_mesh: NavigationMesh, source_geometry_data: NavigationMeshSourceGeometryData3D, callback: Callable = Callable())

void	free_rid(rid: RID)
bool	get_debug_enabled() const
Array[RID]	get_maps() const
int	get_process_info(process_info: ProcessInfo) const
bool	is_baking_navigation_mesh(navigation_mesh: NavigationMesh) const
RID	link_create()
bool	link_get_enabled(link: RID) const
Vector3	link_get_end_position(link: RID) const
float	link_get_enter_cost(link: RID) const
RID	link_get_map(link: RID) const
int	link_get_navigation_layers(link: RID) const
int	link_get_owner_id(link: RID) const
Vector3	link_get_start_position(link: RID) const
float	link_get_travel_cost(link: RID) const
bool	link_is_bidirectional(link: RID) const

void	link_set_bidirectional(link: RID, bidirectional: bool)
void	link_set_enabled(link: RID, enabled: bool)
void	link_set_end_position(link: RID, position: Vector3)
void	link_set_enter_cost(link: RID, enter_cost: float)
void	link_set_map(link: RID, map: RID)
void	link_set_navigation_layers(link: RID, navigation_layers: int)
void	link_set_owner_id(link: RID, owner_id: int)
void	link_set_start_position(link: RID, position: Vector3)
void	link_set_travel_cost(link: RID, travel_cost: float)
RID	map_create()
void	map_force_update(map: RID)
Array[RID]	map_get_agents(map: RID) const
float	map_get_cell_height(map: RID) const
float	map_get_cell_size(map: RID) const
Vector3	map_get_closest_point(map: RID, to_point: Vector3) const

Vector3	map_get_closest_point_normal(map: RID, to_point: Vector3) const
RID	map_get_closest_point_owner(map: RID, to_point: Vector3) const
Vector3	map_get_closest_point_to_segment(map: RID, start: Vector3, end: Vector3, use_collision: bool = false) const
float	map_get_edge_connection_margin(map: RID) const
int	map_get_iteration_id(map: RID) const
float	map_get_link_connection_radius(map: RID) const
Array[RID]	map_get_links(map: RID) const
float	map_get_merge_rasterizer_cell_scale(map: RID) const
Array[RID]	map_get_obstacles(map: RID) const
PackedVector3Array	map_get_path(map: RID, origin: Vector3, destination: Vector3, optimize: bool, navigation_layers: int = 1) const
Vector3	map_get_random_point(map: RID, navigation_layers: int, uniformly: bool) const
Array[RID]	map_get_regions(map: RID) const
Vector3	map_get_up(map: RID) const

bool	map_get_use_edge_connections(map: RID) const
bool	map_is_active(map: RID) const
void	map_set_active(map: RID, active: bool)
void	map_set_cell_height(map: RID, cell_height: float)
void	map_set_cell_size(map: RID, cell_size: float)
void	map_set_edge_connection_margin(map: RID, margin: float)
void	map_set_link_connection_radius(map: RID, radius: float)
void	map_set_merge_rasterizer_cell_scale(map: RID, scale: float)
void	map_set_up(map: RID, up: Vector3)
void	map_set_use_edge_connections(map: RID, enabled: bool)
RID	obstacle_create()
bool	obstacle_get_avoidance_enabled(obstacle: RID) const
int	obstacle_get_avoidance_layers(obstacle: RID) const
float	obstacle_get_height(obstacle: RID) const

RID	obstacle_get_map(obstacle: RID) const
bool	obstacle_get_paused(obstacle: RID) const
Vector3	obstacle_get_position(obstacle: RID) const
float	obstacle_get_radius(obstacle: RID) const
bool	obstacle_get_use_3d_avoidance(obstacle: RID) const
Vector3	obstacle_get_velocity(obstacle: RID) const
PackedVector3Array	obstacle_get_vertices(obstacle: RID) const
void	obstacle_set_avoidance_enabled(obstacle: RID, enabled: bool)
void	obstacle_set_avoidance_layers(obstacle: RID, layers: int)
void	obstacle_set_height(obstacle: RID, height: float)
void	obstacle_set_map(obstacle: RID, map: RID)
void	obstacle_set_paused(obstacle: RID, paused: bool)
void	obstacle_set_position(obstacle: RID, position: Vector3)
void	obstacle_set_radius(obstacle: RID, radius: float)

void	obstacle_set_use_3d_avoidance(obstacle: RID, enabled: bool)
void	obstacle_set_velocity(obstacle: RID, velocity: Vector3)
void	obstacle_set_vertices(obstacle: RID, vertices: PackedVector3Array)
void	parse_source_geometry_data(navigation_mesh: NavigationMesh, source_geometry_data: NavigationMeshSourceGeometryData3D, root_node: Node, callback: Callable = Callable())
void	query_path(parameters: NavigationPathQueryParameters3D, result: NavigationPathQueryResult3D) const
void	region_bake_navigation_mesh(navigation_mesh: NavigationMesh, root_node: Node)
RID	region_create()
Vector3	region_get_connection_pathway_end(region: RID, connection: int) const
Vector3	region_get_connection_pathway_start(region: RID, connection: int) const
int	region_get_connections_count(region: RID) const
bool	region_get_enabled(region: RID) const
float	region_get_enter_cost(region: RID) const

RID	region_get_map(region: RID) const
int	region_get_navigation_layers(region: RID) const
int	region_get_owner_id(region: RID) const
Vector3	region_get_random_point(region: RID, navigation_layers: int, uniformly: bool) const
Transform3D	region_get_transform(region: RID) const
float	region_get_travel_cost(region: RID) const
bool	region_get_use_edge_connections(region: RID) const
bool	region_owns_point(region: RID, point: Vector3) const
void	region_set_enabled(region: RID, enabled: bool)
void	region_set_enter_cost(region: RID, enter_cost: float)
void	region_set_map(region: RID, map: RID)
void	region_set_navigation_layers(region: RID, navigation_layers: int)
void	region_set_navigation_mesh(region: RID, navigation_mesh: NavigationMesh)
void	region_set_owner_id(region: RID, owner_id: int)

void	region_set_transform(region: RID, transform: Transform3D)
void	region_set_travel_cost(region: RID, travel_cost: float)
void	region_set_use_edge_connections(region: RID, enabled: bool)
void	set_active(active: bool)
void	set_debug_enabled(enabled: bool)
PackedVector3Array	simplify_path(path: PackedVector3Array, epsilon: float)
RID	source_geometry_parser_create()
void	source_geometry_parser_set_callback(parser: RID, callback: Callable)

Signals

avoidance_debug_changed() 🔗

Emitted when avoidance debug settings are changed. Only available in debug builds.

map_changed(map: RID) 🔗

Emitted when a navigation map is updated, when a region moves or is modified.

navigation_debug_changed() 🔗

Emitted when navigation debug settings are changed. Only available in debug builds.

Enumerations

enum **ProcessInfo**: 🔗

ProcessInfo **INFO_ACTIVE_MAPS** = `0`

Constant to get the number of active navigation maps.

ProcessInfo **INFO_REGION_COUNT** = `1`

Constant to get the number of active navigation regions.

ProcessInfo **INFO_AGENT_COUNT** = `2`

Constant to get the number of active navigation agents processing avoidance.

ProcessInfo **INFO_LINK_COUNT** = `3`

Constant to get the number of active navigation links.

ProcessInfo **INFO_POLYGON_COUNT** = `4`

Constant to get the number of navigation mesh polygons.

ProcessInfo **INFO_EDGE_COUNT** = `5`

Constant to get the number of navigation mesh polygon edges.

ProcessInfo **INFO_EDGE_MERGE_COUNT** = `6`

Constant to get the number of navigation mesh polygon edges that were merged due to edge key overlap.

ProcessInfo **INFO_EDGE_CONNECTION_COUNT** = `7`

Constant to get the number of navigation mesh polygon edges that are considered connected by edge proximity.

ProcessInfo **INFO_EDGE_FREE_COUNT** = `8`

Constant to get the number of navigation mesh polygon edges that could not be merged but may be still connected by edge proximity or with links.

Method Descriptions

RID **agent_create**()

Creates the agent.

bool **agent_get_avoidance_enabled**(agent: RID) const

Returns `true` if the provided `agent` has avoidance enabled.

int **agent_get_avoidance_layers**(agent: RID) const

Returns the `avoidance_layers` bitmask of the specified `agent`.

int **agent_get_avoidance_mask**(agent: RID) const

Returns the `avoidance_mask` bitmask of the specified `agent`.

float **agent_get_avoidance_priority**(agent: RID) const

Returns the `avoidance_priority` of the specified `agent`.

float **agent_get_height**(agent: RID) const

Returns the `height` of the specified `agent`.

RID **agent_get_map**(agent: RID) const

Returns the navigation map RID the requested `agent` is currently assigned to.

int **agent_get_max_neighbors**(agent: RID) const

Returns the maximum number of other agents the specified `agent` takes into account in the navigation.

---

float **agent_get_max_speed**(agent: RID) const

Returns the maximum speed of the specified `agent`.

---

float **agent_get_neighbor_distance**(agent: RID) const

Returns the maximum distance to other agents the specified `agent` takes into account in the navigation.

---

bool **agent_get_paused**(agent: RID) const

Returns `true` if the specified `agent` is paused.

Vector3 **agent_get_position**(agent: RID) const

Returns the position of the specified `agent` in world space.

---

float **agent_get_radius**(agent: RID) const

Returns the radius of the specified `agent`.

---

float **agent_get_time_horizon_agents**(agent: RID) const

Returns the minimal amount of time for which the specified `agent`'s velocities that are computed by the simulation are safe with respect to other agents.

---

float **agent_get_time_horizon_obstacles**(agent: RID) const

Returns the minimal amount of time for which the specified `agent`'s velocities that are computed by the simulation are safe with respect to static avoidance obstacles.

---

bool **agent_get_use_3d_avoidance**(agent: RID) const

Returns `true` if the provided `agent` uses avoidance in 3D space Vector3(x,y,z) instead of horizontal 2D Vector2(x,y) / Vector3(x,0.0,z).

Vector3 **agent_get_velocity**(agent: RID) const

Returns the velocity of the specified `agent`.

---

bool **agent_has_avoidance_callback**(agent: RID) const

Return `true` if the specified `agent` has an avoidance callback.

---

bool **agent_is_map_changed**(agent: RID) const

Returns true if the map got changed the previous frame.

---

void **agent_set_avoidance_callback**(agent: RID, callback: Callable)

Sets the callback Callable that gets called after each avoidance processing step for the `agent`. The calculated `safe_velocity` will be dispatched with a signal to the object just before the physics calculations.

Note: Created callbacks are always processed independently of the SceneTree state as long as the agent is on a navigation map and not freed. To disable the dispatch of a callback from an agent use agent_set_avoidance_callback again with an empty Callable.

---

void **agent_set_avoidance_enabled**(agent: RID, enabled: bool)

If `enabled` is `true`, the provided `agent` calculates avoidance.

---

void **agent_set_avoidance_layers**(agent: RID, layers: int)

Set the agent's `avoidance_layers` bitmask.

---

void **agent_set_avoidance_mask**(agent: RID, mask: int)

Set the agent's `avoidance_mask` bitmask.

---

void **agent_set_avoidance_priority**(agent: RID, priority: float)

Set the agent's `avoidance_priority` with a `priority` between 0.0 (lowest priority) to 1.0 (highest priority).

The specified `agent` does not adjust the velocity for other agents that would match the `avoidance_mask` but have a lower `avoidance_priority`. This in turn makes the other agents with lower priority adjust their velocities even more to avoid collision with this agent.

---

void **agent_set_height**(agent: RID, height: float)

Updates the provided `agent` `height`.

---

void **agent_set_map**(agent: RID, map: RID)

Puts the agent in the map.

---

void **agent_set_max_neighbors**(agent: RID, count: int)

Sets the maximum number of other agents the agent takes into account in the navigation. The larger this number, the longer the running time of the simulation. If the number is too low, the simulation will not be safe.

---

void **agent_set_max_speed**(agent: RID, max_speed: float)

Sets the maximum speed of the agent. Must be positive.

---

void **agent_set_neighbor_distance**(agent: RID, distance: float)

Sets the maximum distance to other agents this agent takes into account in the navigation. The larger this number, the longer the running time of the simulation. If the number is too low, the simulation will not be safe.

void **agent_set_paused**(agent: RID, paused: bool)

If `paused` is true the specified `agent` will not be processed, e.g. calculate avoidance velocities or receive avoidance callbacks.

void **agent_set_position**(agent: RID, position: Vector3)

Sets the position of the agent in world space.

void **agent_set_radius**(agent: RID, radius: float)

Sets the radius of the agent.

void **agent_set_time_horizon_agents**(agent: RID, time_horizon: float)

The minimal amount of time for which the agent's velocities that are computed by the simulation are safe with respect to other agents. The larger this number, the sooner this agent will respond to the presence of other agents, but the less freedom this agent has in choosing its velocities. A too high value will slow down agents movement considerably. Must be positive.

void **agent_set_time_horizon_obstacles**(agent: RID, time_horizon: float)

The minimal amount of time for which the agent's velocities that are computed by the simulation are safe with respect to static avoidance obstacles. The larger this number, the sooner this agent will respond to the presence of static avoidance obstacles, but the less freedom this agent has in choosing its velocities. A too high value will slow down agents movement considerably. Must be positive.

void **agent_set_use_3d_avoidance**(agent: RID, enabled: bool)

Sets if the agent uses the 2D avoidance or the 3D avoidance while avoidance is enabled.

If `true` the agent calculates avoidance velocities in 3D for the xyz-axis, e.g. for games that take place in air, underwater or space. The 3D using agent only avoids other 3D avoidance using agent's. The 3D using agent only reacts to radius based avoidance obstacles. The 3D using agent ignores any vertices based obstacles. The 3D using agent only avoids other 3D using agent's.

If `false` the agent calculates avoidance velocities in 2D along the xz-axis ignoring the y-axis. The 2D using agent only avoids other 2D avoidance using agent's. The 2D using agent reacts to radius avoidance obstacles. The 2D using agent reacts to vertices based avoidance obstacles. The 2D using agent only avoids other 2D using agent's. 2D using agents will ignore other 2D using agents or obstacles that are below their current position or above their current position including the agents height in 2D avoidance.

---

void **agent_set_velocity**(agent: RID, velocity: Vector3) 🔗

Sets `velocity` as the new wanted velocity for the specified `agent`. The avoidance simulation will try to fulfill this velocity if possible but will modify it to avoid collision with other agent's and obstacles. When an agent is teleported to a new position use agent_set_velocity_forced as well to reset the internal simulation velocity.

---

void **agent_set_velocity_forced**(agent: RID, velocity: Vector3) 🔗

Replaces the internal velocity in the collision avoidance simulation with `velocity` for the specified `agent`. When an agent is teleported to a new position this function should be used in the same frame. If called frequently this function can get agents stuck.

---

void **bake_from_source_geometry_data**(navigation_mesh: NavigationMesh, source_geometry_data: NavigationMeshSourceGeometryData3D, callback: Callable = Callable()) 🔗

Bakes the provided `navigation_mesh` with the data from the provided `source_geometry_data`. After the process is finished the optional `callback` will be called.

---

void **bake_from_source_geometry_data_async**(navigation_mesh: NavigationMesh, source_geometry_data: NavigationMeshSourceGeometryData3D, callback: Callable = Callable()) 🔗

Bakes the provided `navigation_mesh` with the data from the provided `source_geometry_data` as an async task running on a background thread. After the process is finished the optional `callback` will be called.

---

void **free_rid**(rid: RID)

Destroys the given RID.

---

bool **get_debug_enabled**() const

Returns `true` when the NavigationServer has debug enabled.

Array[RID] **get_maps**() const

Returns all created navigation map RIDs on the NavigationServer. This returns both 2D and 3D created navigation maps as there is technically no distinction between them.

---

int **get_process_info**(process_info: ProcessInfo) const

Returns information about the current state of the NavigationServer. See ProcessInfo for a list of available states.

---

bool **is_baking_navigation_mesh**(navigation_mesh: NavigationMesh) const

Returns `true` when the provided navigation mesh is being baked on a background thread.

RID **link_create**()

Create a new link between two positions on a map.

---

bool **link_get_enabled**(link: RID) const

Returns `true` if the specified `link` is enabled.

Vector3 **link_get_end_position**(link: RID) const

Returns the ending position of this `link`.

---

float **link_get_enter_cost**(link: RID) const

Returns the enter cost of this `link`.

RID **link_get_map**(link: RID) const

Returns the navigation map RID the requested `link` is currently assigned to.

---

int **link_get_navigation_layers**(link: RID) const

Returns the navigation layers for this `link`.

---

int **link_get_owner_id**(link: RID) const

Returns the `ObjectID` of the object which manages this link.

Vector3 **link_get_start_position**(link: RID) const

Returns the starting position of this `link`.

---

float **link_get_travel_cost**(link: RID) const

Returns the travel cost of this `link`.

---

bool **link_is_bidirectional**(link: RID) const

Returns whether this `link` can be travelled in both directions.

---

void **link_set_bidirectional**(link: RID, bidirectional: bool)

Sets whether this `link` can be travelled in both directions.

---

void **link_set_enabled**(link: RID, enabled: bool)

If `enabled` is `true`, the specified `link` will contribute to its current navigation map.

---

void **link_set_end_position**(link: RID, position: Vector3)

Sets the exit position for the `link`.

---

void **link_set_enter_cost**(link: RID, enter_cost: float)

Sets the `enter_cost` for this `link`.

---

void **link_set_map**(link: RID, map: RID)

Sets the navigation map RID for the link.

---

void **link_set_navigation_layers**(link: RID, navigation_layers: int)

Set the links's navigation layers. This allows selecting links from a path request (when using map_get_path).

---

void **link_set_owner_id**(link: RID, owner_id: int)

Set the `ObjectID` of the object which manages this link.

---

void **link_set_start_position**(link: RID, position: Vector3)

Sets the entry position for this `link`.

---

void **link_set_travel_cost**(link: RID, travel_cost: float)

Sets the `travel_cost` for this `link`.

RID **map_create**()

Create a new map.

void **map_force_update**(map: RID)

This function immediately forces synchronization of the specified navigation `map` RID. By default navigation maps are only synchronized at the end of each physics frame. This function can be used to immediately (re)calculate all the navigation meshes and region connections of the navigation map. This makes it possible to query a navigation path for a changed map immediately and in the same frame (multiple times if needed).

Due to technical restrictions the current NavigationServer command queue will be flushed. This means all already queued update commands for this physics frame will be executed, even those intended for other maps, regions and agents not part of the specified map. The expensive computation of the navigation meshes and region connections of a map will only be done for the specified map. Other maps will receive the normal synchronization at the end of the physics frame. Should the specified map receive changes after the forced update it will update again as well when the other maps receive their update.

Avoidance processing and dispatch of the `safe_velocity` signals is unaffected by this function and continues to happen for all maps and agents at the end of the physics frame.

Note: With great power comes great responsibility. This function should only be used by users that really know what they are doing and have a good reason for it. Forcing an immediate update of a navigation map requires locking the NavigationServer and flushing the entire NavigationServer command queue. Not only can this severely impact the performance of a game but it can also introduce bugs if used inappropriately without much foresight.

Array[RID] **map_get_agents**(map: RID) const

Returns all navigation agents RIDs that are currently assigned to the requested navigation `map`.

float **map_get_cell_height**(map: RID) const

Returns the map cell height used to rasterize the navigation mesh vertices on the Y axis.

---

float **map_get_cell_size**(map: RID) const 🔗

Returns the map cell size used to rasterize the navigation mesh vertices on the XZ plane.

Vector3 **map_get_closest_point**(map: RID, to_point: Vector3) const 🔗

Returns the point closest to the provided `to_point` on the navigation mesh surface.

Vector3 **map_get_closest_point_normal**(map: RID, to_point: Vector3) const 🔗

Returns the normal for the point returned by map_get_closest_point.

RID **map_get_closest_point_owner**(map: RID, to_point: Vector3) const 🔗

Returns the owner region RID for the point returned by map_get_closest_point.

Vector3 **map_get_closest_point_to_segment**(map: RID, start: Vector3, end: Vector3, use_collision: bool = false) const 🔗

Returns the closest point between the navigation surface and the segment.

---

float **map_get_edge_connection_margin**(map: RID) const 🔗

Returns the edge connection margin of the map. This distance is the minimum vertex distance needed to connect two edges from different regions.

---

int **map_get_iteration_id**(map: RID) const 🔗

Returns the current iteration id of the navigation map. Every time the navigation map changes and synchronizes the iteration id increases. An iteration id of 0 means the navigation map has never synchronized.

Note: The iteration id will wrap back to 1 after reaching its range limit.

---

float **map_get_link_connection_radius**(map: RID) const

Returns the link connection radius of the map. This distance is the maximum range any link will search for navigation mesh polygons to connect to.

Array[RID] **map_get_links**(map: RID) const

Returns all navigation link RIDs that are currently assigned to the requested navigation `map`.

---

float **map_get_merge_rasterizer_cell_scale**(map: RID) const

Returns map's internal merge rasterizer cell scale.

Array[RID] **map_get_obstacles**(map: RID) const

Returns all navigation obstacle RIDs that are currently assigned to the requested navigation `map`.

PackedVector3Array **map_get_path**(map: RID, origin: Vector3, destination: Vector3, optimize: bool, navigation_layers: int = 1) const

Returns the navigation path to reach the destination from the origin. `navigation_layers` is a bitmask of all region navigation layers that are allowed to be in the path.

Vector3 **map_get_random_point**(map: RID, navigation_layers: int, uniformly: bool) const

Returns a random position picked from all map region polygons with matching `navigation_layers`.

If `uniformly` is `true`, all map regions, polygons, and faces are weighted by their surface area (slower).

If `uniformly` is `false`, just a random region and a random polygon are picked (faster).

Array[RID] **map_get_regions**(map: RID) const

Returns all navigation regions RIDs that are currently assigned to the requested navigation `map`.

Vector3 **map_get_up**(map: RID) const

Returns the map's up direction.

bool **map_get_use_edge_connections**(map: RID) const

Returns true if the navigation `map` allows navigation regions to use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin.

bool **map_is_active**(map: RID) const

Returns true if the map is active.

void **map_set_active**(map: RID, active: bool)

Sets the map active.

void **map_set_cell_height**(map: RID, cell_height: float)

Sets the map cell height used to rasterize the navigation mesh vertices on the Y axis. Must match with the cell height of the used navigation meshes.

void **map_set_cell_size**(map: RID, cell_size: float)

Sets the map cell size used to rasterize the navigation mesh vertices on the XZ plane. Must match with the cell size of the used navigation meshes.

void **map_set_edge_connection_margin**(map: RID, margin: float) 🔗

Set the map edge connection margin used to weld the compatible region edges.

---

void **map_set_link_connection_radius**(map: RID, radius: float) 🔗

Set the map's link connection radius used to connect links to navigation polygons.

---

void **map_set_merge_rasterizer_cell_scale**(map: RID, scale: float) 🔗

Set the map's internal merge rasterizer cell scale used to control merging sensitivity.

---

void **map_set_up**(map: RID, up: Vector3) 🔗

Sets the map up direction.

---

void **map_set_use_edge_connections**(map: RID, enabled: bool) 🔗

Set the navigation `map` edge connection use. If `enabled` is `true`, the navigation map allows navigation regions to use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin.

RID **obstacle_create**() 🔗

Creates a new obstacle.

---

bool **obstacle_get_avoidance_enabled**(obstacle: RID) const 🔗

Returns `true` if the provided `obstacle` has avoidance enabled.

---

int **obstacle_get_avoidance_layers**(obstacle: RID) const 🔗

Returns the `avoidance_layers` bitmask of the specified `obstacle`.

---

float **obstacle_get_height**(obstacle: RID) const

Returns the `height` of the specified `obstacle`.

RID **obstacle_get_map**(obstacle: RID) const

Returns the navigation map RID the requested `obstacle` is currently assigned to.

bool **obstacle_get_paused**(obstacle: RID) const

Returns `true` if the specified `obstacle` is paused.

Vector3 **obstacle_get_position**(obstacle: RID) const

Returns the position of the specified `obstacle` in world space.

float **obstacle_get_radius**(obstacle: RID) const

Returns the radius of the specified dynamic `obstacle`.

bool **obstacle_get_use_3d_avoidance**(obstacle: RID) const

Returns `true` if the provided `obstacle` uses avoidance in 3D space Vector3(x,y,z) instead of horizontal 2D Vector2(x,y) / Vector3(x,0.0,z).

Vector3 **obstacle_get_velocity**(obstacle: RID) const

Returns the velocity of the specified dynamic `obstacle`.

PackedVector3Array **obstacle_get_vertices**(obstacle: RID) const

Returns the outline vertices for the specified `obstacle`.

void **obstacle_set_avoidance_enabled**(obstacle: RID, enabled: bool)

If `enabled` is `true`, the provided `obstacle` affects avoidance using agents.

---

void **obstacle_set_avoidance_layers**(obstacle: RID, layers: int)

Set the obstacles's `avoidance_layers` bitmask.

---

void **obstacle_set_height**(obstacle: RID, height: float)

Sets the `height` for the `obstacle`. In 3D agents will ignore obstacles that are above or below them while using 2D avoidance.

---

void **obstacle_set_map**(obstacle: RID, map: RID)

Assigns the `obstacle` to a navigation map.

---

void **obstacle_set_paused**(obstacle: RID, paused: bool)

If `paused` is true the specified `obstacle` will not be processed, e.g. affect avoidance velocities.

---

void **obstacle_set_position**(obstacle: RID, position: Vector3)

Updates the `position` in world space for the `obstacle`.

---

void **obstacle_set_radius**(obstacle: RID, radius: float)

Sets the radius of the dynamic obstacle.

---

void **obstacle_set_use_3d_avoidance**(obstacle: RID, enabled: bool)

Sets if the `obstacle` uses the 2D avoidance or the 3D avoidance while avoidance is enabled.

---

void **obstacle_set_velocity**(obstacle: RID, velocity: Vector3)

Sets `velocity` of the dynamic `obstacle`. Allows other agents to better predict the movement of the dynamic obstacle. Only works in combination with the radius of the obstacle.

---

void **obstacle_set_vertices**(obstacle: RID, vertices: PackedVector3Array)

Sets the outline vertices for the obstacle. If the vertices are winded in clockwise order agents will be pushed in by the obstacle, else they will be pushed out.

---

void **parse_source_geometry_data**(navigation_mesh: NavigationMesh, source_geometry_data: NavigationMeshSourceGeometryData3D, root_node: Node, callback: Callable = Callable())

Parses the SceneTree for source geometry according to the properties of `navigation_mesh`. Updates the provided `source_geometry_data` resource with the resulting data. The resource can then be used to bake a navigation mesh with bake_from_source_geometry_data. After the process is finished the optional `callback` will be called.

Note: This function needs to run on the main thread or with a deferred call as the SceneTree is not thread-safe.

Performance: While convenient, reading data arrays from Mesh resources can affect the frame rate negatively. The data needs to be received from the GPU, stalling the RenderingServer in the process. For performance prefer the use of e.g. collision shapes or creating the data arrays entirely in code.

---

void **query_path**(parameters: NavigationPathQueryParameters3D, result: NavigationPathQueryResult3D) const

Queries a path in a given navigation map. Start and target position and other parameters are defined through NavigationPathQueryParameters3D. Updates the provided NavigationPathQueryResult3D result object with the path among other results requested by the query.

---

void **region_bake_navigation_mesh**(navigation_mesh: NavigationMesh, root_node: Node)

Deprecated: This method is deprecated due to core threading changes. To upgrade existing code, first create a NavigationMeshSourceGeometryData3D resource. Use this resource with parse_source_geometry_data to parse the SceneTree for nodes that should contribute to the navigation mesh baking. The SceneTree parsing needs to happen on the main thread. After the parsing is finished use the resource with bake_from_source_geometry_data to bake a navigation mesh.

Bakes the `navigation_mesh` with bake source geometry collected starting from the `root_node`.

RID **region_create**()

Creates a new region.

Vector3 **region_get_connection_pathway_end**(region: RID, connection: int) const

Returns the ending point of a connection door. `connection` is an index between 0 and the return value of region_get_connections_count.

Vector3 **region_get_connection_pathway_start**(region: RID, connection: int) const

Returns the starting point of a connection door. `connection` is an index between 0 and the return value of region_get_connections_count.

---

int **region_get_connections_count**(region: RID) const

Returns how many connections this `region` has with other regions in the map.

---

bool **region_get_enabled**(region: RID) const

Returns `true` if the specified `region` is enabled.

---

float **region_get_enter_cost**(region: RID) const

Returns the enter cost of this `region`.

RID **region_get_map**(region: RID) const

Returns the navigation map RID the requested `region` is currently assigned to.

---

int **region_get_navigation_layers**(region: RID) const

Returns the region's navigation layers.

---

int **region_get_owner_id**(region: RID) const

Returns the `ObjectID` of the object which manages this region.

Vector3 **region_get_random_point**(region: RID, navigation_layers: int, uniformly: bool) const

Returns a random position picked from all region polygons with matching `navigation_layers`.

If `uniformly` is `true`, all region polygons and faces are weighted by their surface area (slower).

If `uniformly` is `false`, just a random polygon and face is picked (faster).

Transform3D **region_get_transform**(region: RID) const

Returns the global transformation of this `region`.

---

float **region_get_travel_cost**(region: RID) const

Returns the travel cost of this `region`.

---

bool **region_get_use_edge_connections**(region: RID) const

Returns true if the navigation `region` is set to use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin.

---

bool **region_owns_point**(region: RID, point: Vector3) const

Returns `true` if the provided `point` in world space is currently owned by the provided navigation `region`. Owned in this context means that one of the region's navigation mesh polygon faces has a possible position at the closest distance to this point compared to all other navigation meshes from other navigation regions that are also registered on the navigation map of the provided region.

If multiple navigation meshes have positions at equal distance the navigation region whose polygons are processed first wins the ownership. Polygons are processed in the same order that navigation regions were registered on the NavigationServer.

Note: If navigation meshes from different navigation regions overlap (which should be avoided in general) the result might not be what is expected.

---

void **region_set_enabled**(region: RID, enabled: bool)

If `enabled` is `true`, the specified `region` will contribute to its current navigation map.

---

void **region_set_enter_cost**(region: RID, enter_cost: float)

Sets the `enter_cost` for this `region`.

---

void **region_set_map**(region: RID, map: RID)

Sets the map for the region.

---

void **region_set_navigation_layers**(region: RID, navigation_layers: int)

Set the region's navigation layers. This allows selecting regions from a path request (when using map_get_path).

---

void **region_set_navigation_mesh**(region: RID, navigation_mesh: NavigationMesh) 🔗

Sets the navigation mesh for the region.

---

void **region_set_owner_id**(region: RID, owner_id: int) 🔗

Set the `ObjectID` of the object which manages this region.

---

void **region_set_transform**(region: RID, transform: Transform3D) 🔗

Sets the global transformation for the region.

---

void **region_set_travel_cost**(region: RID, travel_cost: float) 🔗

Sets the `travel_cost` for this `region`.

---

void **region_set_use_edge_connections**(region: RID, enabled: bool) 🔗

If `enabled` is `true`, the navigation `region` will use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin.

---

void **set_active**(active: bool) 🔗

Control activation of this server.

---

void **set_debug_enabled**(enabled: bool) 🔗

If `true` enables debug mode on the NavigationServer.

PackedVector3Array **simplify_path**(path: PackedVector3Array, epsilon: float) 🔗

Returns a simplified version of `path` with less critical path points removed. The simplification amount is in worlds units and controlled by `epsilon`. The simplification uses a variant of Ramer-Douglas-Peucker algorithm for curve point decimation.

Path simplification can be helpful to mitigate various path following issues that can arise with certain agent types and script behaviors. E.g. "steering" agents or avoidance in "open fields".

RID **source_geometry_parser_create**() 🔗

Creates a new source geometry parser. If a Callable is set for the parser with source_geometry_parser_set_callback the callback will be called for every single node that gets parsed whenever parse_source_geometry_data is used.

---

void **source_geometry_parser_set_callback**(parser: RID, callback: Callable) 🔗

Sets the `callback` Callable for the specific source geometry `parser`. The Callable will receive a call with the following parameters:

- `navigation_mesh` - The NavigationMesh reference used to define the parse settings. Do NOT edit or add directly to the navigation mesh.
- `source_geometry_data` - The NavigationMeshSourceGeometryData3D reference. Add custom source geometry for navigation mesh baking to this object.
- `node` - The Node that is parsed.

Node3DGizmo

Inherits: RefCounted **<** Object

Inherited By: EditorNode3DGizmo

Abstract class to expose editor gizmos for Node3D.

Description

This abstract class helps connect the Node3D scene with the editor-specific EditorNode3DGizmo class.

Node3DGizmo by itself has no exposed API, refer to Node3D.add_gizmo and pass it an EditorNode3DGizmo instance.

OfflineMultiplayerPeer

Inherits: MultiplayerPeer **<** PacketPeer **<** RefCounted **<** Object

A MultiplayerPeer which is always connected and acts as a server.

Description

This is the default MultiplayerAPI.multiplayer_peer for the Node.multiplayer. It mimics the behavior of a server with no peers connected.

This means that the SceneTree will act as the multiplayer authority by default. Calls to MultiplayerAPI.is_server will return `true`, and calls to MultiplayerAPI.get_unique_id will return MultiplayerPeer.TARGET_PEER_SERVER.

OggPacketSequencePlayback

Inherits: RefCounted **<** Object

There is currently no description for this class. Please help us by contributing one!

OpenXRAPIExtension

Inherits: RefCounted < Object

Makes the OpenXR API available for GDExtension.

Description

OpenXRAPIExtension makes OpenXR available for GDExtension. It provides the OpenXR API to GDExtension through the get_instance_proc_addr method, and the OpenXR instance through get_instance.

It also provides methods for querying the status of OpenXR initialization, and helper methods for ease of use of the API with GDExtension.

Tutorials

- XrResult documentation [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrResult.html]
- XrInstance documentation [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrInstance.html]
- XrSpace documentation [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrSpace.html]
- XrSession documentation [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrSession.html]
- XrSystemId documentation [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrSystemId.html]
- xrBeginSession documentation [https://registry.khronos.org/OpenXR/specs/1.0/man/html/xrBeginSession.html]
- XrPosef documentation [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrPosef.html]

Methods

bool	can_render()
String	get_error_string(result: int)
int	get_hand_tracker(hand_index: int)
int	get_instance()
int	get_instance_proc_addr(name: String)
int	get_next_frame_time()

int	get_play_space()
int	get_predicted_display_time()
int	get_session()
String	get_swapchain_format_name(swapchain_format: int)
int	get_system_id()
OpenXRAlphaBlendModeSupport	is_environment_blend_mode_alpha_supported()
bool	is_initialized()
bool	is_running()
bool	openxr_is_enabled(check_run_in_editor: bool) static
void	register_composition_layer_provider(extension: OpenXRExtensionWrapperExtension)
void	set_emulate_environment_blend_mode_alpha_blend(enabled: bool)
Transform3D	transform_from_pose(pose: `const void*`)
void	unregister_composition_layer_provider(extension: OpenXRExtensionWrapperExtension)
bool	xr_result(result: int, format: String, args: Array)

Enumerations

enum **OpenXRAlphaBlendModeSupport**: 🔗

OpenXRAlphaBlendModeSupport **OPENXR_ALPHA_BLEND_MODE_SUPPORT_NONE** = `0`

Means that XRInterface.XR_ENV_BLEND_MODE_ALPHA_BLEND isn't supported at all.

OpenXRAlphaBlendModeSupport **OPENXR_ALPHA_BLEND_MODE_SUPPORT_REAL** = `1`

Means that XRInterface.XR_ENV_BLEND_MODE_ALPHA_BLEND is really supported.

OpenXRAlphaBlendModeSupport **OPENXR_ALPHA_BLEND_MODE_SUPPORT_EMULATING** = `2`

Means that XRInterface.XR_ENV_BLEND_MODE_ALPHA_BLEND is emulated.

Method Descriptions

bool **can_render**() 🔗

Returns `true` if OpenXR is initialized for rendering with an XR viewport.

String **get_error_string**(result: int) 🔗

Returns an error string for the given XrResult [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrResult.html].

---

int **get_hand_tracker**(hand_index: int) 🔗

Returns the corresponding `XRHandTrackerEXT` handle for the given hand index value.

---

int **get_instance**() 🔗

Returns the XrInstance [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrInstance.html] created during the initialization of the OpenXR API.

---

int **get_instance_proc_addr**(name: String) 🔗

Returns the function pointer of the OpenXR function with the specified name, cast to an integer. If the function with the given name does not exist, the method returns `0`.

Note: `openxr/util.h` contains utility macros for acquiring OpenXR functions, e.g. `GDEXTENSION_INIT_XR_FUNC_V(xrCreateAction)`.

---

int **get_next_frame_time**() 🔗

Returns the predicted display timing for the next frame.

---

int **get_play_space**() 🔗

Returns the play space, which is an XrSpace [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrSpace.html] cast to an integer.

---

int **get_predicted_display_time**()

Returns the predicted display timing for the current frame.

---

int **get_session**()

Returns the OpenXR session, which is an XrSession [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrSession.html] cast to an integer.

String **get_swapchain_format_name**(swapchain_format: int)

Returns the name of the specified swapchain format.

---

int **get_system_id**()

Returns the id of the system, which is a XrSystemId [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrSystemId.html] cast to an integer.

OpenXRAlphaBlendModeSupport **is_environment_blend_mode_alpha_supported**()

Returns OpenXRAlphaBlendModeSupport denoting if XRInterface.XR_ENV_BLEND_MODE_ALPHA_BLEND is really supported, emulated or not supported at all.

---

bool **is_initialized**()

Returns `true` if OpenXR is initialized.

---

bool **is_running**()

Returns `true` if OpenXR is running (xrBeginSession [https://registry.khronos.org/OpenXR/specs/1.0/man/html/xrBeginSession.html] was successfully called and the swapchains were created).

---

bool **openxr_is_enabled**(check_run_in_editor: bool) static

Returns `true` if OpenXR is enabled.

---

void **register_composition_layer_provider**(extension: OpenXRExtensionWrapperExtension) 🔗

Registers the given extension as a composition layer provider.

---

void **set_emulate_environment_blend_mode_alpha_blend**(enabled: bool) 🔗

If set to `true`, an OpenXR extension is loaded which is capable of emulating the XRInterface.XR_ENV_BLEND_MODE_ALPHA_BLEND blend mode.

Transform3D **transform_from_pose**(pose: `const void*`) 🔗

Creates a Transform3D from an XrPosef [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrPosef.html].

---

void **unregister_composition_layer_provider**(extension: OpenXRExtensionWrapperExtension) 🔗

Unregisters the given extension as a composition layer provider.

---

bool **xr_result**(result: int, format: String, args: Array) 🔗

Returns `true` if the provided XrResult [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrResult.html] (cast to an integer) is successful. Otherwise returns `false` and prints the XrResult [https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrResult.html] converted to a string, with the specified additional information.

OpenXRExtensionWrapperExtension

Inherits: Object

Allows clients to implement OpenXR extensions with GDExtension.

Description

OpenXRExtensionWrapperExtension allows clients to implement OpenXR extensions with GDExtension. The extension should be registered with register_extension_wrapper.

Methods

int	_get_composition_layer(index: int) virtual
int	_get_composition_layer_count() virtual
int	_get_composition_layer_order(index: int) virtual
Dictionary	_get_requested_extensions() virtual
PackedStringArray	_get_suggested_tracker_names() virtual
Array[Dictionary]	_get_viewport_composition_layer_extension_properties() virtual
Dictionary	_get_viewport_composition_layer_extension_property_defaults() virtual
void	_on_before_instance_created() virtual
bool	_on_event_polled(event: `const void*`) virtual
void	_on_instance_created(instance: int) virtual

void	_on_instance_destroyed() virtual
void	_on_main_swapchains_created() virtual
void	_on_pre_render() virtual
void	_on_process() virtual
void	_on_register_metadata() virtual
void	_on_session_created(session: int) virtual
void	_on_session_destroyed() virtual
void	_on_state_exiting() virtual
void	_on_state_focused() virtual
void	_on_state_idle() virtual
void	_on_state_loss_pending() virtual
void	_on_state_ready() virtual
void	_on_state_stopping() virtual
void	_on_state_synchronized() virtual
void	_on_state_visible() virtual
void	_on_viewport_composition_layer_destroyed(layer: `const void*`) virtual

int	_set_hand_joint_locations_and_get_next_pointer(hand_index: int, next_pointer: `void*`) virtual
int	_set_instance_create_info_and_get_next_pointer(next_pointer: `void*`) virtual
int	_set_session_create_and_get_next_pointer(next_pointer: `void*`) virtual
int	_set_swapchain_create_info_and_get_next_pointer(next_pointer: `void*`) virtual
int	_set_system_properties_and_get_next_pointer(next_pointer: `void*`) virtual
int	_set_viewport_composition_layer_and_get_next_pointer(layer: `const void*`, property_values: Dictionary, next_pointer: `void*`) virtual
OpenXRAPIExtension	get_openxr_api()
void	register_extension_wrapper()

Method Descriptions

int **_get_composition_layer**(index: int) virtual 🔗

Returns a pointer to an `XrCompositionLayerBaseHeader` struct to provide the given composition layer.

This will only be called if the extension previously registered itself with OpenXRAPIExtension.register_composition_layer_provider.

---

int **_get_composition_layer_count**() virtual 🔗

Returns the number of composition layers this extension wrapper provides via _get_composition_layer.

This will only be called if the extension previously registered itself with OpenXRAPIExtension.register_composition_layer_provider.

---

int **_get_composition_layer_order**(index: int) virtual 🔗

Returns an integer that will be used to sort the given composition layer provided via _get_composition_layer. Lower numbers will move the layer to the front of the list, and higher numbers to the end. The default projection layer has an order of `0`, so layers provided by this method should probably be above or below (but not exactly) `0`.

This will only be called if the extension previously registered itself with OpenXRAPIExtension.register_composition_layer_provider.

Dictionary **_get_requested_extensions**() virtual 🔗

Returns a Dictionary of OpenXR extensions related to this extension. The Dictionary should contain the name of the extension, mapped to a `bool *` cast to an integer:

- If the `bool *` is a `nullptr` this extension is mandatory.
- If the `bool *` points to a boolean, the boolean will be updated to `true` if the extension is enabled.

PackedStringArray **_get_suggested_tracker_names**() virtual 🔗

Returns a PackedStringArray of positional tracker names that are used within the extension wrapper.

Array[Dictionary] **_get_viewport_composition_layer_extension_properties**() virtual 🔗

Gets an array of Dictionarys that represent properties, just like Object._get_property_list, that will be added to OpenXRCompositionLayer nodes.

Dictionary **_get_viewport_composition_layer_extension_property_defaults**() virtual 🔗

Gets a Dictionary containing the default values for the properties returned by _get_viewport_composition_layer_extension_properties.

---

void **_on_before_instance_created**() virtual

Called before the OpenXR instance is created.

---

bool **_on_event_polled**(event: `const void*`) virtual

Called when there is an OpenXR event to process. When implementing, return `true` if the event was handled, return `false` otherwise.

---

void **_on_instance_created**(instance: int) virtual

Called right after the OpenXR instance is created.

---

void **_on_instance_destroyed**() virtual

Called right before the OpenXR instance is destroyed.

---

void **_on_main_swapchains_created**() virtual

Called right after the main swapchains are (re)created.

---

void **_on_pre_render**() virtual

Called right before the XR viewports begin their rendering step.

---

void **_on_process**() virtual

Called as part of the OpenXR process handling. This happens right before general and physics processing steps of the main loop. During this step controller data is queried and made available to game logic.

---

void **_on_register_metadata**() virtual

Allows extensions to register additional controller metadata. This function is called even when the OpenXR API is not constructed as the metadata needs to be available to the editor.

Extensions should also provide metadata regardless of whether they are supported on the host system. The controller data is used to setup action maps for users who may have access to the relevant hardware.

---

void **_on_session_created**(session: int) virtual

Called right after the OpenXR session is created.

---

void **_on_session_destroyed**() virtual

Called right before the OpenXR session is destroyed.

---

void **_on_state_exiting**() virtual

Called when the OpenXR session state is changed to exiting.

---

void **_on_state_focused**() virtual

Called when the OpenXR session state is changed to focused. This state is the active state when the game runs.

---

void **_on_state_idle**() virtual

Called when the OpenXR session state is changed to idle.

---

void **_on_state_loss_pending**() virtual

Called when the OpenXR session state is changed to loss pending.

---

void **_on_state_ready**() virtual

Called when the OpenXR session state is changed to ready. This means OpenXR is ready to set up the session.

---

void **_on_state_stopping**() virtual

Called when the OpenXR session state is changed to stopping.

---

void **_on_state_synchronized**() virtual

Called when the OpenXR session state is changed to synchronized. OpenXR also returns to this state when the application loses focus.

void **_on_state_visible**() virtual

Called when the OpenXR session state is changed to visible. This means OpenXR is now ready to receive frames.

void **_on_viewport_composition_layer_destroyed**(layer: `const void*`) virtual

Called when a composition layer created via OpenXRCompositionLayer is destroyed.

`layer` is a pointer to an `XrCompositionLayerBaseHeader` struct.

int **_set_hand_joint_locations_and_get_next_pointer**(hand_index: int, next_pointer: `void*`) virtual

Adds additional data structures when each hand tracker is created.

int **_set_instance_create_info_and_get_next_pointer**(next_pointer: `void*`) virtual

Adds additional data structures when the OpenXR instance is created.

int **_set_session_create_and_get_next_pointer**(next_pointer: `void*`) virtual

Adds additional data structures when the OpenXR session is created.

int **_set_swapchain_create_info_and_get_next_pointer**(next_pointer: `void*`) virtual

Adds additional data structures when creating OpenXR swapchains.

int **_set_system_properties_and_get_next_pointer**(next_pointer: `void*`) virtual

Adds additional data structures when interogating OpenXR system abilities.

int **_set_viewport_composition_layer_and_get_next_pointer**(layer: `const void*`, property_values: Dictionary, next_pointer: `void*`) virtual

Adds additional data structures to composition layers created by OpenXRCompositionLayer.

`property_values` contains the values of the properties returned by _get_viewport_composition_layer_extension_properties.

`layer` is a pointer to an `XrCompositionLayerBaseHeader` struct.

OpenXRAPIExtension **get_openxr_api**()

Returns the created OpenXRAPIExtension, which can be used to access the OpenXR API.

---

void **register_extension_wrapper**()

Registers the extension. This should happen at core module initialization level.

OpenXRInteractionProfileMetadata

Inherits: Object

Meta class registering supported devices in OpenXR.

Description

This class allows OpenXR core and extensions to register metadata relating to supported interaction devices such as controllers, trackers, haptic devices, etc. It is primarily used by the action map editor and to sanitize any action map by removing extension-dependent entries when applicable.

Methods

void	register_interaction_profile(display_name: String, openxr_path: String, openxr_extension_name: String)
void	register_io_path(interaction_profile: String, display_name: String, toplevel_path: String, openxr_path: String, openxr_extension_name: String, action_type: ActionType)
void	register_profile_rename(old_name: String, new_name: String)

void	register_top_level_path(display_name: String, openxr_path: String, openxr_extension_name: String)

Method Descriptions

void **register_interaction_profile**(display_name: String, openxr_path: String, openxr_extension_name: String) 🔗

Registers an interaction profile using its OpenXR designation (e.g. `/interaction_profiles/khr/simple_controller` is the profile for OpenXR's simple controller profile).

`display_name` is the description shown to the user. `openxr_path` is the interaction profile path being registered. `openxr_extension_name` optionally restricts this profile to the given extension being enabled/available. If the extension is not available, the profile and all related entries used in an action map are filtered out.

void **register_io_path**(interaction_profile: String, display_name: String, toplevel_path: String, openxr_path: String, openxr_extension_name: String, action_type: ActionType) 🔗

Registers an input/output path for the given `interaction_profile`. The profile should previously have been registered using register_interaction_profile. `display_name` is the description shown to the user. `toplevel_path` specifies the bind path this input/output can be bound to (e.g. `/user/hand/left` or `/user/hand/right`). `openxr_path` is the action input/output being registered (e.g. `/user/hand/left/input/aim/pose`). `openxr_extension_name` restricts this input/output to an enabled/available extension, this doesn't need to repeat the extension on the profile but relates to overlapping extension (e.g. `XR_EXT_palm_pose` that introduces `…/input/palm_ext/pose` input paths). `action_type` defines the type of input or output provided by OpenXR.

---

void **register_profile_rename**(old_name: String, new_name: String) 🔗

Allows for renaming old interaction profile paths to new paths to maintain backwards compatibility with older action maps.

---

void **register_top_level_path**(display_name: String, openxr_path: String, openxr_extension_name: String) 🔗

Registers a top level path to which profiles can be bound. For instance `/user/hand/left` refers to the bind point for the player's left hand. Extensions can register additional top level paths, for instance a haptic vest extension might register `/user/body/vest`.

`display_name` is the name shown to the user. `openxr_path` is the top level path being registered. `openxr_extension_name` is optional and ensures the top level path is only used if the specified extension is available/enabled.

When a top level path ends up being bound by OpenXR, a XRPositionalTracker is instantiated to manage the state of the device.

OpenXRInterface

Inherits: XRInterface **<** RefCounted **<** Object

Our OpenXR interface.

Description

The OpenXR interface allows Godot to interact with OpenXR runtimes and make it possible to create XR experiences and games.

Due to the needs of OpenXR this interface works slightly different than other plugin based XR interfaces. It needs to be initialized when Godot starts. You need to enable OpenXR, settings for this can be found in your games project settings under the XR heading. You do need to mark a viewport for use with XR in order for Godot to know which render result should be output to the headset.

Tutorials

- Setting up XR

Properties

float	display_refresh_rate	`0.0`
bool	foveation_dynamic	`false`

int	foveation_level	`0`
float	render_target_size_multiplier	`1.0`
float	vrs_min_radius	`20.0`
float	vrs_strength	`1.0`

Methods

Array	get_action_sets() const
Array	get_available_display_refresh_rates() const
Vector3	get_hand_joint_angular_velocity(hand: Hand, joint: HandJoints) const
BitField[HandJointFlags]	get_hand_joint_flags(hand: Hand, joint: HandJoints) const
Vector3	get_hand_joint_linear_velocity(hand: Hand, joint: HandJoints) const

Vector3	get_hand_joint_position(hand: Hand, joint: HandJoints) const
float	get_hand_joint_radius(hand: Hand, joint: HandJoints) const
Quaternion	get_hand_joint_rotation(hand: Hand, joint: HandJoints) const
HandTrackedSource	get_hand_tracking_source(hand: Hand) const
HandMotionRange	get_motion_range(hand: Hand) const
bool	is_action_set_active(name: String) const
bool	is_eye_gaze_interaction_supported()
bool	is_foveation_supported() const
bool	is_hand_interaction_supported() const
bool	is_hand_tracking_supported()
void	set_action_set_active(name: String, active: bool)

void	set_motion_range(hand: Hand, motion_range: HandMotionRange)

Signals

instance_exiting()

Informs our OpenXR instance is exiting.

---

pose_recentered()

Informs the user queued a recenter of the player position.

---

refresh_rate_changed(refresh_rate: float)

Informs the user the HMD refresh rate has changed.

Note: Only emitted if XR runtime supports the refresh rate extension.

---

session_begun()

Informs our OpenXR session has been started.

---

session_focussed()

Informs our OpenXR session now has focus.

session_loss_pending()

Informs our OpenXR session is in the process of being lost.

---

session_stopping()

Informs our OpenXR session is stopping.

---

session_visible()

Informs our OpenXR session is now visible (output is being sent to the HMD).

Enumerations

enum **Hand**:

Hand **HAND_LEFT** = `0`

Left hand.

Hand **HAND_RIGHT** = `1`

Right hand.

Hand **HAND_MAX** = `2`

Maximum value for the hand enum.

---

enum **HandMotionRange**:

HandMotionRange **HAND_MOTION_RANGE_UNOBSTRUCTED** = `0`

Full hand range, if user closes their hands, we make a full fist.

HandMotionRange **HAND_MOTION_RANGE_CONFORM_TO_CONTROLLER** = `1`

Conform to controller, if user closes their hands, the tracked data conforms to the shape of the controller.

HandMotionRange **HAND_MOTION_RANGE_MAX** = `2`

Maximum value for the motion range enum.

---

enum **HandTrackedSource**: 🔗

HandTrackedSource **HAND_TRACKED_SOURCE_UNKNOWN** = `0`

The source of hand tracking data is unknown (the extension is likely unsupported).

HandTrackedSource **HAND_TRACKED_SOURCE_UNOBSTRUCTED** = `1`

The source of hand tracking is unobstructed, this means that an accurate method of hand tracking is used, e.g. optical hand tracking, data gloves, etc.

HandTrackedSource **HAND_TRACKED_SOURCE_CONTROLLER** = `2`

The source of hand tracking is a controller, bone positions are inferred from controller inputs.

HandTrackedSource **HAND_TRACKED_SOURCE_MAX** = `3`

Maximum value for the hand tracked source enum.

---

enum **HandJoints**: 🔗

HandJoints **HAND_JOINT_PALM** = `0`

Palm joint.

HandJoints **HAND_JOINT_WRIST** = `1`

Wrist joint.

HandJoints **HAND_JOINT_THUMB_METACARPAL** = `2`

Thumb metacarpal joint.

HandJoints **HAND_JOINT_THUMB_PROXIMAL** = `3`

Thumb proximal joint.

HandJoints **HAND_JOINT_THUMB_DISTAL** = `4`

Thumb distal joint.

HandJoints **HAND_JOINT_THUMB_TIP** = `5`

Thumb tip joint.

HandJoints **HAND_JOINT_INDEX_METACARPAL** = `6`

Index metacarpal joint.

HandJoints **HAND_JOINT_INDEX_PROXIMAL** = `7`

Index proximal joint.

HandJoints **HAND_JOINT_INDEX_INTERMEDIATE** = `8`

Index intermediate joint.

HandJoints **HAND_JOINT_INDEX_DISTAL** = `9`

Index distal joint.

HandJoints **HAND_JOINT_INDEX_TIP** = `10`

Index tip joint.

HandJoints **HAND_JOINT_MIDDLE_METACARPAL** = `11`

Middle metacarpal joint.

HandJoints **HAND_JOINT_MIDDLE_PROXIMAL** = `12`

Middle proximal joint.

HandJoints **HAND_JOINT_MIDDLE_INTERMEDIATE** = `13`

Middle intermediate joint.

HandJoints **HAND_JOINT_MIDDLE_DISTAL** = `14`

Middle distal joint.

HandJoints **HAND_JOINT_MIDDLE_TIP** = `15`

Middle tip joint.

HandJoints **HAND_JOINT_RING_METACARPAL** = `16`

Ring metacarpal joint.

HandJoints **HAND_JOINT_RING_PROXIMAL** = `17`

Ring proximal joint.

HandJoints **HAND_JOINT_RING_INTERMEDIATE** = `18`

Ring intermediate joint.

HandJoints **HAND_JOINT_RING_DISTAL** = `19`

Ring distal joint.

HandJoints **HAND_JOINT_RING_TIP** = `20`

Ring tip joint.

HandJoints **HAND_JOINT_LITTLE_METACARPAL** = `21`

Little metacarpal joint.

HandJoints **HAND_JOINT_LITTLE_PROXIMAL** = `22`

Little proximal joint.

HandJoints **HAND_JOINT_LITTLE_INTERMEDIATE** = `23`

Little intermediate joint.

HandJoints **HAND_JOINT_LITTLE_DISTAL** = `24`

Little distal joint.

HandJoints **HAND_JOINT_LITTLE_TIP** = `25`

Little tip joint.

HandJoints **HAND_JOINT_MAX** = `26`

Maximum value for the hand joint enum.

---

flags **HandJointFlags**: 🔗

HandJointFlags **HAND_JOINT_NONE** = `0`

No flags are set.

HandJointFlags **HAND_JOINT_ORIENTATION_VALID** = `1`

If set, the orientation data is valid, otherwise, the orientation data is unreliable and should not be used.

HandJointFlags **HAND_JOINT_ORIENTATION_TRACKED** = `2`

If set, the orientation data comes from tracking data, otherwise, the orientation data contains predicted data.

HandJointFlags **HAND_JOINT_POSITION_VALID** = `4`

If set, the positional data is valid, otherwise, the positional data is unreliable and should not be used.

HandJointFlags **HAND_JOINT_POSITION_TRACKED** = `8`

If set, the positional data comes from tracking data, otherwise, the positional data contains predicted data.

HandJointFlags **HAND_JOINT_LINEAR_VELOCITY_VALID** = `16`

If set, our linear velocity data is valid, otherwise, the linear velocity data is unreliable and should not be used.

HandJointFlags **HAND_JOINT_ANGULAR_VELOCITY_VALID** = `32`

If set, our angular velocity data is valid, otherwise, the angular velocity data is unreliable and should not be used.

Property Descriptions

float **display_refresh_rate** = `0.0` 🔗

- void **set_display_refresh_rate**(value: float)
- float **get_display_refresh_rate**()

The display refresh rate for the current HMD. Only functional if this feature is supported by the OpenXR runtime and after the interface has been initialized.

---

bool **foveation_dynamic** = `false` 🔗

- void **set_foveation_dynamic**(value: bool)
- bool **get_foveation_dynamic**()

Enable dynamic foveation adjustment, the interface must be initialized before this is accessible. If enabled foveation will automatically adjusted between low and foveation_level.

Note: Only works on compatibility renderer.

---

int **foveation_level** = `0` 🔗

- void **set_foveation_level**(value: int)
- int **get_foveation_level**()

Set foveation level from 0 (off) to 3 (high), the interface must be initialized before this is accessible.

Note: Only works on compatibility renderer.

---

float **render_target_size_multiplier** = `1.0` 🔗

- void **set_render_target_size_multiplier**(value: float)
- float **get_render_target_size_multiplier**()

The render size multiplier for the current HMD. Must be set before the interface has been initialized.

---

float **vrs_min_radius** = `20.0` 🔗

- void **set_vrs_min_radius**(value: float)
- float **get_vrs_min_radius**()

The minimum radius around the focal point where full quality is guaranteed if VRS is used as a percentage of screen size.

Note: Mobile and Forward+ renderers only. Requires Viewport.vrs_mode to be set to Viewport.VRS_XR.

---

float **vrs_strength** = `1.0` 🔗

- void **set_vrs_strength**(value: float)
- float **get_vrs_strength**()

The strength used to calculate the VRS density map. The greater this value, the more noticeable VRS is. This improves performance at the cost of quality.

Note: Mobile and Forward+ renderers only. Requires Viewport.vrs_mode to be set to Viewport.VRS_XR.

Method Descriptions

Array **get_action_sets**() const 🔗

Returns a list of action sets registered with Godot (loaded from the action map at runtime).

Array **get_available_display_refresh_rates**() const 🔗

Returns display refresh rates supported by the current HMD. Only returned if this feature is supported by the OpenXR runtime and after the interface has been initialized.

Vector3 **get_hand_joint_angular_velocity**(hand: Hand, joint: HandJoints) const 🔗

Deprecated: Use XRHandTracker.get_hand_joint_angular_velocity obtained from XRServer.get_tracker instead.

If handtracking is enabled, returns the angular velocity of a joint (`joint`) of a hand (`hand`) as provided by OpenXR. This is relative to XROrigin3D!

BitField[HandJointFlags] **get_hand_joint_flags**(hand: Hand, joint: HandJoints) const 🔗

Deprecated: Use XRHandTracker.get_hand_joint_flags obtained from XRServer.get_tracker instead.

If handtracking is enabled, returns flags that inform us of the validity of the tracking data.

Vector3 **get_hand_joint_linear_velocity**(hand: Hand, joint: HandJoints) const 🔗

Deprecated: Use XRHandTracker.get_hand_joint_linear_velocity obtained from XRServer.get_tracker instead.

If handtracking is enabled, returns the linear velocity of a joint (`joint`) of a hand (`hand`) as provided by OpenXR. This is relative to XROrigin3D without worldscale applied!

Vector3 **get_hand_joint_position**(hand: Hand, joint: HandJoints) const 🔗

Deprecated: Use XRHandTracker.get_hand_joint_transform obtained from XRServer.get_tracker instead.

If handtracking is enabled, returns the position of a joint (`joint`) of a hand (`hand`) as provided by OpenXR. This is relative to XROrigin3D without worldscale applied!

---

float **get_hand_joint_radius**(hand: Hand, joint: HandJoints) const 🔗

Deprecated: Use XRHandTracker.get_hand_joint_radius obtained from XRServer.get_tracker instead.

If handtracking is enabled, returns the radius of a joint (`joint`) of a hand (`hand`) as provided by OpenXR. This is without worldscale applied!

Quaternion **get_hand_joint_rotation**(hand: Hand, joint: HandJoints) const 🔗

Deprecated: Use XRHandTracker.get_hand_joint_transform obtained from XRServer.get_tracker instead.

If handtracking is enabled, returns the rotation of a joint (`joint`) of a hand (`hand`) as provided by OpenXR.

HandTrackedSource **get_hand_tracking_source**(hand: Hand) const 🔗

Deprecated: Use XRHandTracker.hand_tracking_source obtained from XRServer.get_tracker instead.

If handtracking is enabled and hand tracking source is supported, gets the source of the hand tracking data for `hand`.

HandMotionRange **get_motion_range**(hand: Hand) const

If handtracking is enabled and motion range is supported, gets the currently configured motion range for `hand`.

---

bool **is_action_set_active**(name: String) const

Returns `true` if the given action set is active.

---

bool **is_eye_gaze_interaction_supported**()

Returns the capabilities of the eye gaze interaction extension.

Note: This only returns a valid value after OpenXR has been initialized.

---

bool **is_foveation_supported**() const

Returns `true` if OpenXR's foveation extension is supported, the interface must be initialized before this returns a valid value.

Note: This feature is only available on the compatibility renderer and currently only available on some stand alone headsets. For Vulkan set Viewport.vrs_mode to `VRS_XR` on desktop.

---

bool **is_hand_interaction_supported**() const

Returns `true` if OpenXR's hand interaction profile is supported and enabled.

Note: This only returns a valid value after OpenXR has been initialized.

---

bool **is_hand_tracking_supported**() 🔗

Returns `true` if OpenXR's hand tracking is supported and enabled.

Note: This only returns a valid value after OpenXR has been initialized.

---

void **set_action_set_active**(name: String, active: bool) 🔗

Sets the given action set as active or inactive.

---

void **set_motion_range**(hand: Hand, motion_range: HandMotionRange) 🔗

If handtracking is enabled and motion range is supported, sets the currently configured motion range for `hand` to `motion_range`.

OS

Inherits: Object

Provides access to common operating system functionalities.

Description

The **OS** class wraps the most common functionalities for communicating with the host operating system, such as the video driver, delays, environment variables, execution of binaries, command line, etc.

Note: In Godot 4, **OS** functions related to window management, clipboard, and TTS were moved to the DisplayServer singleton (and the Window class). Functions related to time were removed and are only available in the Time class.

Tutorials

- Operating System Testing Demo [https://godotengine.org/asset-library/asset/2789]

Properties

bool	delta_smoothing	`true`
bool	low_processor_usage_mode	`false`

int	low_processor_usage_mode_sleep_usec	`6900`

Methods

void	alert(text: String, title: String = "Alert!")
void	close_midi_inputs()
void	crash(message: String)
int	create_instance(arguments: PackedStringArray)
int	create_process(path: String, arguments: PackedStringArray, open_console: bool = false)
void	delay_msec(msec: int) const
void	delay_usec(usec: int) const
int	execute(path: String, arguments: PackedStringArray, output: Array = [], read_stderr: bool = false, open_console: bool = false)

Dictionary	execute_with_pipe(path: String, arguments: PackedStringArray)
Key	find_keycode_from_string(string: String) const
String	get_cache_dir() const
PackedStringArray	get_cmdline_args()
PackedStringArray	get_cmdline_user_args()
String	get_config_dir() const
PackedStringArray	get_connected_midi_inputs()
String	get_data_dir() const
String	get_distribution_name() const
PackedByteArray	get_entropy(size: int)
String	get_environment(variable: String) const
String	get_executable_path() const

PackedStringArray	get_granted_permissions() const
String	get_keycode_string(code: Key) const
String	get_locale() const
String	get_locale_language() const
int	get_main_thread_id() const
Dictionary	get_memory_info() const
String	get_model_name() const
String	get_name() const
int	get_process_exit_code(pid: int) const
int	get_process_id() const
int	get_processor_count() const
String	get_processor_name() const
PackedStringArray	get_restart_on_exit_arguments() const

int	get_static_memory_peak_usage() const
int	get_static_memory_usage() const
String	get_system_ca_certificates()
String	get_system_dir(dir: SystemDir, shared_storage: bool = true) const
String	get_system_font_path(font_name: String, weight: int = 400, stretch: int = 100, italic: bool = false) const
PackedStringArray	get_system_font_path_for_text(font_name: String, text: String, locale: String = "", script: String = "", weight: int = 400, stretch: int = 100, italic: bool = false) const
PackedStringArray	get_system_fonts() const
int	get_thread_caller_id() const
String	get_unique_id() const
String	get_user_data_dir() const

String	get_version() const
PackedStringArray	get_video_adapter_driver_info() const
bool	has_environment(variable: String) const
bool	has_feature(tag_name: String) const
bool	is_debug_build() const
bool	is_keycode_unicode(code: int) const
bool	is_process_running(pid: int) const
bool	is_restart_on_exit_set() const
bool	is_sandboxed() const
bool	is_stdout_verbose() const
bool	is_userfs_persistent() const
Error	kill(pid: int)
Error	move_to_trash(path: String) const

void	open_midi_inputs()
String	read_string_from_stdin()
bool	request_permission(name: String)
bool	request_permissions()
void	revoke_granted_permissions()
void	set_environment(variable: String, value: String) const
void	set_restart_on_exit(restart: bool, arguments: PackedStringArray = PackedStringArray())
Error	set_thread_name(name: String)
void	set_use_file_access_save_and_swap(enabled: bool)
Error	shell_open(uri: String)
Error	shell_show_in_file_manager(file_or_dir_path: String, open_folder: bool = true)

void	unset_environment(variable: String) const

Enumerations

enum **RenderingDriver**: 🔗

RenderingDriver **RENDERING_DRIVER_VULKAN** = `0`

The Vulkan rendering driver. It requires Vulkan 1.0 support and automatically uses features from Vulkan 1.1 and 1.2 if available.

RenderingDriver **RENDERING_DRIVER_OPENGL3** = `1`

The OpenGL 3 rendering driver. It uses OpenGL 3.3 Core Profile on desktop platforms, OpenGL ES 3.0 on mobile devices, and WebGL 2.0 on Web.

RenderingDriver **RENDERING_DRIVER_D3D12** = `2`

The Direct3D 12 rendering driver.

---

enum **SystemDir**: 🔗

SystemDir **SYSTEM_DIR_DESKTOP** = `0`

Refers to the Desktop directory path.

SystemDir **SYSTEM_DIR_DCIM** = `1`

Refers to the DCIM (Digital Camera Images) directory path.

SystemDir **SYSTEM_DIR_DOCUMENTS** = `2`

Refers to the Documents directory path.

SystemDir **SYSTEM_DIR_DOWNLOADS** = `3`

Refers to the Downloads directory path.

SystemDir **SYSTEM_DIR_MOVIES** = `4`

Refers to the Movies (or Videos) directory path.

SystemDir **SYSTEM_DIR_MUSIC** = `5`

Refers to the Music directory path.

SystemDir **SYSTEM_DIR_PICTURES** = `6`

Refers to the Pictures directory path.

SystemDir **SYSTEM_DIR_RINGTONES** = `7`

Refers to the Ringtones directory path.

Property Descriptions

bool **delta_smoothing** = `true` 🔗

- void **set_delta_smoothing**(value: bool)
- bool **is_delta_smoothing_enabled**()

If `true`, the engine filters the time delta measured between each frame, and attempts to compensate for random variation. This only works on systems where V-Sync is active.

Note: On start-up, this is the same as ProjectSettings.application/run/delta_smoothing.

---

bool **low_processor_usage_mode** = `false`

- void **set_low_processor_usage_mode**(value: bool)
- bool **is_in_low_processor_usage_mode**()

If `true`, the engine optimizes for low processor usage by only refreshing the screen if needed. Can improve battery consumption on mobile.

Note: On start-up, this is the same as ProjectSettings.application/run/low_processor_mode.

---

int **low_processor_usage_mode_sleep_usec** = `6900`

- void **set_low_processor_usage_mode_sleep_usec**(value: int)
- int **get_low_processor_usage_mode_sleep_usec**()

The amount of sleeping between frames when the low-processor usage mode is enabled, in microseconds. Higher values will result in lower CPU usage. See also low_processor_usage_mode.

Note: On start-up, this is the same as ProjectSettings.application/run/low_processor_mode_sleep_usec.

Method Descriptions

void **alert**(text: String, title: String = "Alert!")

Displays a modal dialog box using the host platform's implementation. The engine execution is blocked until the dialog is closed.

---

void **close_midi_inputs**() 🔗

Shuts down the system MIDI driver. Godot will no longer receive InputEventMIDI. See also open_midi_inputs and get_connected_midi_inputs.

Note: This method is implemented on Linux, macOS and Windows.

---

void **crash**(message: String) 🔗

Crashes the engine (or the editor if called within a `@tool` script). See also kill.

Note: This method should *only* be used for testing the system's crash handler, not for any other purpose. For general error reporting, use (in order of preference) @GDScript.assert, @GlobalScope.push_error, or alert.

---

int **create_instance**(arguments: PackedStringArray) 🔗

Creates a new instance of Godot that runs independently. The `arguments` are used in the given order and separated by a space.

If the process is successfully created, this method returns the new process' ID, which you can use to monitor the process (and potentially terminate it with kill). If the process cannot be created, this method returns `-1`.

See create_process if you wish to run a different process.

Note: This method is implemented on Android, Linux, macOS and Windows.

---

int **create_process**(path: String, arguments: PackedStringArray, open_console: bool = false) 🔗

Creates a new process that runs independently of Godot. It will not terminate when Godot terminates. The path specified in `path` must exist and be an executable file or macOS `.app` bundle. The path is resolved based on the current platform. The `arguments` are used in the given order and separated by a space.

On Windows, if `open_console` is `true` and the process is a console app, a new terminal window will be opened.

If the process is successfully created, this method returns its process ID, which you can use to monitor the process (and potentially terminate it with kill). Otherwise, this method returns `-1`.

For example, running another instance of the project:

GDScript

```
var pid = OS.create_process(OS.get_executable_path(), [])
```

C#

```
var pid = OS.CreateProcess(OS.GetExecutablePath(), new string[]
{});
```

See execute if you wish to run an external command and retrieve the results.

Note: This method is implemented on Android, Linux, macOS, and Windows.

Note: On macOS, sandboxed applications are limited to run only embedded helper executables, specified during export or system .app bundle, system .app bundles will ignore arguments.

---

void **delay_msec**(msec: int) const

Delays execution of the current thread by `msec` milliseconds. `msec` must be greater than or equal to `0`. Otherwise, delay_msec does nothing and prints an error message.

Note: delay_msec is a *blocking* way to delay code execution. To delay code execution in a non-blocking way, you may use SceneTree.create_timer. Awaiting with SceneTreeTimer delays the execution of code placed below the `await` without affecting the rest of the project (or editor, for EditorPlugins and EditorScripts).

Note: When delay_msec is called on the main thread, it will freeze the project and will prevent it from redrawing and registering input until the delay has passed. When using delay_msec as part of an EditorPlugin or EditorScript, it will freeze the editor but won't freeze the project if it is currently running (since the project is an independent child process).

---

void **delay_usec**(usec: int) const

Delays execution of the current thread by `usec` microseconds. `usec` must be greater than or equal to `0`. Otherwise, delay_usec does nothing and prints an error message.

Note: delay_usec is a *blocking* way to delay code execution. To delay code execution in a non-blocking way, you may use SceneTree.create_timer. Awaiting with a SceneTreeTimer delays the execution of code placed below the `await` without affecting the rest of the project (or editor, for EditorPlugins and EditorScripts).

Note: When delay_usec is called on the main thread, it will freeze the project and will prevent it from redrawing and registering input until the delay has passed. When using delay_usec as part of an EditorPlugin or EditorScript, it will freeze the editor but won't freeze the project if it is currently running (since the project is an independent child process).

---

int **execute**(path: String, arguments: PackedStringArray, output: Array = [], read_stderr: bool = false, open_console: bool = false)

Executes the given process in a *blocking* way. The file specified in `path` must exist and be executable. The system path resolution will be used. The `arguments` are used in the given order, separated by spaces, and wrapped in quotes.

If an `output` array is provided, the complete shell output of the process is appended to `output` as a single String element. If `read_stderr` is `true`, the output to the standard error stream is also appended to the array.

On Windows, if `open_console` is `true` and the process is a console app, a new terminal window is opened.

This method returns the exit code of the command, or `-1` if the process fails to execute.

Note: The main thread will be blocked until the executed command terminates. Use Thread to create a separate thread that will not block the main thread, or use create_process to create a completely independent process.

For example, to retrieve a list of the working directory's contents:

GDScript

```
var output = []
var exit_code = OS.execute("ls", ["-l", "/tmp"], output)
```

C#

```
var output = new Godot.Collections.Array();
int exitCode = OS.Execute("ls", new string[] {"-l", "/tmp"},
output);
```

If you wish to access a shell built-in or execute a composite command, a platform-specific shell can be invoked. For example:

GDScript

```
var output = []
OS.execute("CMD.exe", ["/C", "cd %TEMP% && dir"], output)
```

C#

```
var output = new Godot.Collections.Array();
OS.Execute("CMD.exe", new string[] {"/C", "cd %TEMP% && dir"},
output);
```

Note: This method is implemented on Android, Linux, macOS, and Windows.

Note: To execute a Windows command interpreter built-in command, specify `cmd.exe` in `path`, `/c` as the first argument, and the desired command as the second argument.

Note: To execute a PowerShell built-in command, specify `powershell.exe` in `path`, `-Command` as the first argument, and the desired command as the second argument.

Note: To execute a Unix shell built-in command, specify shell executable name in `path`, `-c` as the first argument, and the desired command as the second argument.

Note: On macOS, sandboxed applications are limited to run only embedded helper executables, specified during export.

Note: On Android, system commands such as `dumpsys` can only be run on a rooted device.

Dictionary **execute_with_pipe**(path: String, arguments: PackedStringArray) 🔗

Creates a new process that runs independently of Godot with redirected IO. It will not terminate when Godot terminates. The path specified in `path` must exist and be an executable file or macOS `.app` bundle. The path is resolved based on the current platform. The `arguments` are used in the given order and separated by a space.

If the process cannot be created, this method returns an empty Dictionary. Otherwise, this method returns a Dictionary with the following keys:

- `"stdio"` - FileAccess to access the process stdin and stdout pipes (read/write).
- `"stderr"` - FileAccess to access the process stderr pipe (read only).
- `"pid"` - Process ID as an int, which you can use to monitor the process (and potentially terminate it with kill).

Note: This method is implemented on Android, Linux, macOS, and Windows.

Note: To execute a Windows command interpreter built-in command, specify `cmd.exe` in `path`, `/c` as the first argument, and the desired command as the second argument.

Note: To execute a PowerShell built-in command, specify `powershell.exe` in `path`, `-Command` as the first argument, and the desired command as the second argument.

Note: To execute a Unix shell built-in command, specify shell executable name in `path`, `-c` as the first argument, and the desired command as the second argument.

Note: On macOS, sandboxed applications are limited to run only embedded helper executables, specified during export or system .app bundle, system .app bundles will ignore arguments.

Key **find_keycode_from_string**(string: String) const

Finds the keycode for the given string. The returned values are equivalent to the Key constants.

GDScript

```
print(OS.find_keycode_from_string("C"))         # Prints 67 (KEY_C)
print(OS.find_keycode_from_string("Escape"))     # Prints 4194305 (KEY_ESCAPE)
print(OS.find_keycode_from_string("Shift+Tab")) # Prints 37748738 (KEY_MASK_SHIFT | KEY_TAB)
print(OS.find_keycode_from_string("Unknown"))    # Prints 0 (KEY_NONE)
```

C#

```
GD.Print(OS.FindKeycodeFromString("C"));          // Prints C (Key.C)
GD.Print(OS.FindKeycodeFromString("Escape"));     // Prints Escape (Key.Escape)
GD.Print(OS.FindKeycodeFromString("Shift+Tab")); // Prints 37748738 (KeyModifierMask.MaskShift | Key.Tab)
GD.Print(OS.FindKeycodeFromString("Unknown"));    // Prints None (Key.None)
```

See also get_keycode_string.

String **get_cache_dir**() const

Returns the *global* cache data directory according to the operating system's standards.

On the Linux/BSD platform, this path can be overridden by setting the `XDG_CACHE_HOME` environment variable before starting the project. See File paths in Godot projects in the documentation for more information. See also get_config_dir and get_data_dir.

Not to be confused with get_user_data_dir, which returns the *project-specific* user data path.

PackedStringArray **get_cmdline_args**()

Returns the command-line arguments passed to the engine.

Command-line arguments can be written in any form, including both `--key value` and `--key=value` forms so they can be properly parsed, as long as custom command-line arguments do not conflict with engine arguments.

You can also incorporate environment variables using the get_environment method.

You can set ProjectSettings.editor/run/main_run_args to define command-line arguments to be passed by the editor when running the project.

Here's a minimal example on how to parse command-line arguments into a Dictionary using the `--key=value` form for arguments:

GDScript

```
var arguments = {}
for argument in OS.get_cmdline_args():
    if argument.contains("="):
        var key_value = argument.split("=")
        arguments[key_value[0].trim_prefix("--")] =
key_value[1]
    else:
        # Options without an argument will be present in the
```

```
dictionary,
        # with the value set to an empty string.
        arguments[argument.trim_prefix("--")] = ""
```

C#

```
var arguments = new Dictionary<string, string>();
foreach (var argument in OS.GetCmdlineArgs())
{
    if (argument.Contains('='))
    {
        string[] keyValue = argument.Split("=");
        arguments[keyValue[0].TrimPrefix("--")] = keyValue[1];
    }
    else
    {
        // Options without an argument will be present in the
dictionary,
        // with the value set to an empty string.
        arguments[argument.TrimPrefix("--")] = "";
    }
}
```

Note: Passing custom user arguments directly is not recommended, as the engine may discard or modify them. Instead, pass the standard UNIX double dash (`--`) and then the custom arguments, which the engine will ignore by design. These can be read via get_cmdline_user_args.

PackedStringArray **get_cmdline_user_args**()

Returns the command-line user arguments passed to the engine. User arguments are ignored by the engine and reserved for the user. They are passed after the double dash `--` argument. `++` may be used when `--` is intercepted by another program (such as `startx`).

```
# Godot has been executed with the following command:
# godot --fullscreen -- --level=2 --hardcore
```

```
OS.get_cmdline_args()      # Returns ["--fullscreen", "--level=2", "--hardcore"]
OS.get_cmdline_user_args() # Returns ["--level=2", "--hardcore"]
```

To get all passed arguments, use get_cmdline_args.

String **get_config_dir**() const

Returns the *global* user configuration directory according to the operating system's standards.

On the Linux/BSD platform, this path can be overridden by setting the `XDG_CONFIG_HOME` environment variable before starting the project. See File paths in Godot projects in the documentation for more information. See also get_cache_dir and get_data_dir.

Not to be confused with get_user_data_dir, which returns the *project-specific* user data path.

PackedStringArray **get_connected_midi_inputs**()

Returns an array of connected MIDI device names, if they exist. Returns an empty array if the system MIDI driver has not previously been initialized with open_midi_inputs. See also close_midi_inputs.

Note: This method is implemented on Linux, macOS and Windows.

String **get_data_dir**() const

Returns the *global* user data directory according to the operating system's standards.

On the Linux/BSD platform, this path can be overridden by setting the `XDG_DATA_HOME` environment variable before starting the project. See File paths in Godot projects in the documentation for more information. See also get_cache_dir and get_config_dir.

Not to be confused with get_user_data_dir, which returns the *project-specific* user data path.

String **get_distribution_name**() const 🔗

Returns the name of the distribution for Linux and BSD platforms (e.g. "Ubuntu", "Manjaro", "OpenBSD", etc.).

Returns the same value as get_name for stock Android ROMs, but attempts to return the custom ROM name for popular Android derivatives such as "LineageOS".

Returns the same value as get_name for other platforms.

Note: This method is not supported on the Web platform. It returns an empty string.

PackedByteArray **get_entropy**(size: int) 🔗

Generates a PackedByteArray of cryptographically secure random bytes with given `size`.

Note: Generating large quantities of bytes using this method can result in locking and entropy of lower quality on most platforms. Using Crypto.generate_random_bytes is preferred in most cases.

String **get_environment**(variable: String) const 🔗

Returns the value of the given environment variable, or an empty string if `variable` doesn't exist.

Note: Double-check the casing of `variable`. Environment variable names are case-sensitive on all platforms except Windows.

Note: On macOS, applications do not have access to shell environment variables.

String **get_executable_path**() const

Returns the file path to the current engine executable.

Note: On macOS, if you want to launch another instance of Godot, always use create_instance instead of relying on the executable path.

PackedStringArray **get_granted_permissions**() const

On Android devices: Returns the list of dangerous permissions that have been granted.

On macOS: Returns the list of user selected folders accessible to the application (sandboxed applications only). Use the native file dialog to request folder access permission.

String **get_keycode_string**(code: Key) const

Returns the given keycode as a String.

GDScript

```
print(OS.get_keycode_string(KEY_C))                          # Prints
"C"
print(OS.get_keycode_string(KEY_ESCAPE))                     # Prints
```

```
"Escape"
print(OS.get_keycode_string(KEY_MASK_SHIFT | KEY_TAB)) # Prints
"Shift+Tab"
```

C#

```
GD.Print(OS.GetKeycodeString(Key.C));
// Prints "C"
GD.Print(OS.GetKeycodeString(Key.Escape));
// Prints "Escape"
GD.Print(OS.GetKeycodeString((Key)KeyModifierMask.MaskShift |
Key.Tab)); // Prints "Shift+Tab"
```

See also find_keycode_from_string, InputEventKey.keycode, and InputEventKey.get_keycode_with_modifiers.

String **get_locale**() const

Returns the host OS locale as a String of the form `language_Script_COUNTRY_VARIANT@extra`. Every substring after `language` is optional and may not exist.

- `language` - 2 or 3-letter language code [https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes], in lower case.
- `Script` - 4-letter script code [https://en.wikipedia.org/wiki/ISO_15924], in title case.
- `COUNTRY` - 2 or 3-letter country code [https://en.wikipedia.org/wiki/ISO_3166-1], in upper case.
- `VARIANT` - language variant, region and sort order. The variant can have any number of underscored keywords.
- `extra` - semicolon separated list of additional key words. This may include currency, calendar, sort order and numbering system information.

If you want only the language code and not the fully specified locale from the OS, you can use get_locale_language.

String **get_locale_language**() const 🔗

Returns the host OS locale's 2 or 3-letter language code [https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes] as a string which should be consistent on all platforms. This is equivalent to extracting the `language` part of the get_locale string.

This can be used to narrow down fully specified locale strings to only the "common" language code, when you don't need the additional information about country code or variants. For example, for a French Canadian user with `fr_CA` locale, this would return `fr`.

---

int **get_main_thread_id**() const 🔗

Returns the ID of the main thread. See get_thread_caller_id.

Note: Thread IDs are not deterministic and may be reused across application restarts.

Dictionary **get_memory_info**() const 🔗

Returns a Dictionary containing information about the current memory with the following entries:

- `"physical"` - total amount of usable physical memory in bytes. This value can be slightly less than the actual physical memory amount, since it does not include memory reserved by the kernel and devices.
- `"free"` - amount of physical memory, that can be immediately allocated without disk access or other costly operations, in bytes. The process might be able to allocate more physical memory, but this action will require moving inactive pages to disk, which can be expensive.

- `"available"` - amount of memory that can be allocated without extending the swap file(s), in bytes. This value includes both physical memory and swap.
- `"stack"` - size of the current thread stack in bytes.

Note: Each entry's value may be `-1` if it is unknown.

String **get_model_name**() const

Returns the model name of the current device.

Note: This method is implemented on Android and iOS. Returns `"GenericDevice"` on unsupported platforms.

String **get_name**() const

Returns the name of the host platform.

- On Windows, this is `"Windows"`.
- On macOS, this is `"macOS"`.
- On Linux-based operating systems, this is `"Linux"`.
- On BSD-based operating systems, this is `"FreeBSD"`, `"NetBSD"`, `"OpenBSD"`, or `"BSD"` as a fallback.
- On Android, this is `"Android"`.
- On iOS, this is `"iOS"`.
- On Web, this is `"Web"`.

Note: Custom builds of the engine may support additional platforms, such as consoles, possibly returning other names.

GDScript

```
match OS.get_name():
    "Windows":
        print("Welcome to Windows!")
    "macOS":
        print("Welcome to macOS!")
    "Linux", "FreeBSD", "NetBSD", "OpenBSD", "BSD":
        print("Welcome to Linux/BSD!")
    "Android":
        print("Welcome to Android!")
    "iOS":
        print("Welcome to iOS!")
    "Web":
        print("Welcome to the Web!")
```

C#

```
switch (OS.GetName())
{
    case "Windows":
        GD.Print("Welcome to Windows");
        break;
    case "macOS":
        GD.Print("Welcome to macOS!");
        break;
    case "Linux":
    case "FreeBSD":
    case "NetBSD":
    case "OpenBSD":
    case "BSD":
        GD.Print("Welcome to Linux/BSD!");
        break;
    case "Android":
        GD.Print("Welcome to Android!");
        break;
    case "iOS":
        GD.Print("Welcome to iOS!");
        break;
    case "Web":
        GD.Print("Welcome to the Web!");
        break;
}
```

Note: On Web platforms, it is still possible to determine the host platform's OS with feature tags. See has_feature.

---

int **get_process_exit_code**(pid: int) const

Returns the exit code of a spawned process once it has finished running (see is_process_running).

Returns `-1` if the `pid` is not a PID of a spawned child process, the process is still running, or the method is not implemented for the current platform.

Note: Returns `-1` if the `pid` is a macOS bundled app process.

Note: This method is implemented on Android, Linux, macOS and Windows.

---

int **get_process_id**() const

Returns the number used by the host machine to uniquely identify this application.

Note: This method is implemented on Android, iOS, Linux, macOS, and Windows.

---

int **get_processor_count**() const

Returns the number of *logical* CPU cores available on the host machine. On CPUs with HyperThreading enabled, this number will be greater than the number of *physical* CPU cores.

String **get_processor_name**() const

Returns the full name of the CPU model on the host machine (e.g. `"Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz"`).

Note: This method is only implemented on Windows, macOS, Linux and iOS. On Android and Web, get_processor_name returns an empty string.

PackedStringArray **get_restart_on_exit_arguments**() const 🔗

Returns the list of command line arguments that will be used when the project automatically restarts using set_restart_on_exit. See also is_restart_on_exit_set.

int **get_static_memory_peak_usage**() const 🔗

Returns the maximum amount of static memory used. Only works in debug builds.

int **get_static_memory_usage**() const 🔗

Returns the amount of static memory being used by the program in bytes. Only works in debug builds.

String **get_system_ca_certificates**() 🔗

Returns the list of certification authorities trusted by the operating system as a string of concatenated certificates in PEM format.

String **get_system_dir**(dir: SystemDir, shared_storage: bool = true) const 🔗

Returns the path to commonly used folders across different platforms, as defined by `dir`. See the SystemDir constants for available locations.

Note: This method is implemented on Android, Linux, macOS and Windows.

Note: Shared storage is implemented on Android and allows to differentiate between app specific and shared directories, if `shared_storage` is `true`. Shared directories have additional restrictions on Android.

String **get_system_font_path**(font_name: String, weight: int = 400, stretch: int = 100, italic: bool = false) const 🔗

Returns the path to the system font file with `font_name` and style. Returns an empty string if no matching fonts found.

The following aliases can be used to request default fonts: "sans-serif", "serif", "monospace", "cursive", and "fantasy".

Note: Returned font might have different style if the requested style is not available.

Note: This method is implemented on Android, iOS, Linux, macOS and Windows.

PackedStringArray **get_system_font_path_for_text**(font_name: String, text: String, locale: String = "", script: String = "", weight: int = 400, stretch: int = 100, italic: bool = false) const 🔗

Returns an array of the system substitute font file paths, which are similar to the font with `font_name` and style for the specified text, locale, and script. Returns an empty array if no matching fonts found.

The following aliases can be used to request default fonts: "sans-serif", "serif", "monospace", "cursive", and "fantasy".

Note: Depending on OS, it's not guaranteed that any of the returned fonts will be suitable for rendering specified text. Fonts should be loaded and checked in the order they are returned, and the first suitable one used.

Note: Returned fonts might have different style if the requested style is not available or belong to a different font family.

Note: This method is implemented on Android, iOS, Linux, macOS and Windows.

PackedStringArray **get_system_fonts**() const 🔗

Returns the list of font family names available.

Note: This method is implemented on Android, iOS, Linux, macOS and Windows.

int **get_thread_caller_id**() const 🔗

Returns the ID of the current thread. This can be used in logs to ease debugging of multi-threaded applications.

Note: Thread IDs are not deterministic and may be reused across application restarts.

String **get_unique_id**() const 🔗

Returns a string that is unique to the device.

Note: This string may change without notice if the user reinstalls their operating system, upgrades it, or modifies their hardware. This means it should generally not be used to encrypt persistent data, as the data saved before an unexpected ID change would become inaccessible. The returned string may also be falsified using external programs, so do not rely on the string returned by this method for security purposes.

Note: On Web, returns an empty string and generates an error, as this method cannot be implemented for security reasons.

String **get_user_data_dir**() const 🔗

Returns the absolute directory path where user data is written (the `user://` directory in Godot). The path depends on the project name and ProjectSettings.application/config/use_custom_user_dir.

- On Windows, this is `%AppData%\Godot\app_userdata\[project_name]`, or `%AppData%\[custom_name]` if `use_custom_user_dir` is set. `%AppData%` expands to `%UserProfile%\AppData\Roaming`.
- On macOS, this is `~/Library/Application Support/Godot/app_userdata/[project_name]`, or `~/Library/Application Support/[custom_name]` if `use_custom_user_dir` is set.
- On Linux and BSD, this is `~/.local/share/godot/app_userdata/[project_name]`, or `~/.local/share/[custom_name]` if `use_custom_user_dir` is set.
- On Android and iOS, this is a sandboxed directory in either internal or external storage, depending on the user's configuration.
- On Web, this is a virtual directory managed by the browser.

If the project name is empty, `[project_name]` falls back to `[unnamed project]`.

Not to be confused with get_data_dir, which returns the *global* (non-project-specific) user home directory.

String **get_version**() const 🔗

Returns the exact production and build version of the operating system. This is different from the branded version used in marketing. This helps to distinguish between different releases of operating systems, including minor versions, and insider and custom builds.

- For Windows, the major and minor version are returned, as well as the build number. For example, the returned string may look like `10.0.9926` for a build of Windows 10, and it may look like `6.1.7601` for a build of Windows 7 SP1.
- For rolling distributions, such as Arch Linux, an empty string is returned.
- For macOS and iOS, the major and minor version are returned, as well as the patch number.
- For Android, the SDK version and the incremental build number are returned. If it's a custom ROM, it attempts to return its version instead.

Note: This method is not supported on the Web platform. It returns an empty string.

PackedStringArray **get_video_adapter_driver_info**() const 🔗

Returns the video adapter driver name and version for the user's currently active graphics card, as a PackedStringArray. See also RenderingServer.get_video_adapter_api_version.

The first element holds the driver name, such as `nvidia`, `amdgpu`, etc.

The second element holds the driver version. For example, on the `nvidia` driver on a Linux/BSD platform, the version is in the format `510.85.02`. For Windows, the driver's format is `31.0.15.1659`.

Note: This method is only supported on Linux/BSD and Windows when not running in headless mode. On other platforms, it returns an empty array.

---

bool **has_environment**(variable: String) const

Returns `true` if the environment variable with the name `variable` exists.

Note: Double-check the casing of `variable`. Environment variable names are case-sensitive on all platforms except Windows.

---

bool **has_feature**(tag_name: String) const

Returns `true` if the feature for the given feature tag is supported in the currently running instance, depending on the platform, build, etc. Can be used to check whether you're currently running a debug build, on a certain platform or arch, etc. Refer to the Feature Tags documentation for more details.

Note: Tag names are case-sensitive.

Note: On the Web platform, one of the following additional tags is defined to indicate the host platform: `web_android`, `web_ios`, `web_linuxbsd`, `web_macos`, or `web_windows`.

---

bool **is_debug_build**() const

Returns `true` if the Godot binary used to run the project is a *debug* export template, or when running in the editor.

Returns `false` if the Godot binary used to run the project is a *release* export template.

Note: To check whether the Godot binary used to run the project is an export template (debug or release), use `OS.has_feature("template")` instead.

---

bool **is_keycode_unicode**(code: int) const

Returns `true` if the input keycode corresponds to a Unicode character. For a list of codes, see the Key constants.

GDScript

```
print(OS.is_keycode_unicode(KEY_G))      # Prints true
print(OS.is_keycode_unicode(KEY_KP_4))   # Prints true
print(OS.is_keycode_unicode(KEY_TAB))    # Prints false
print(OS.is_keycode_unicode(KEY_ESCAPE)) # Prints false
```

C#

```
GD.Print(OS.IsKeycodeUnicode((long)Key.G));      // Prints true
GD.Print(OS.IsKeycodeUnicode((long)Key.Kp4));    // Prints true
GD.Print(OS.IsKeycodeUnicode((long)Key.Tab));    // Prints
false
GD.Print(OS.IsKeycodeUnicode((long)Key.Escape)); // Prints
false
```

---

bool **is_process_running**(pid: int) const

Returns `true` if the child process ID (`pid`) is still running or `false` if it has terminated. `pid` must be a valid ID generated from create_process.

Note: This method is implemented on Android, iOS, Linux, macOS, and Windows.

---

bool **is_restart_on_exit_set**() const

Returns `true` if the project will automatically restart when it exits for any reason, `false` otherwise. See also set_restart_on_exit and get_restart_on_exit_arguments.

---

bool **is_sandboxed**() const 🔗

Returns `true` if the application is running in the sandbox.

Note: This method is only implemented on macOS and Linux.

---

bool **is_stdout_verbose**() const 🔗

Returns `true` if the engine was executed with the `--verbose` or `-v` command line argument, or if ProjectSettings.debug/settings/stdout/verbose_stdout is `true`. See also @GlobalScope.print_verbose.

---

bool **is_userfs_persistent**() const 🔗

Returns `true` if the `user://` file system is persistent, that is, its state is the same after a player quits and starts the game again. Relevant to the Web platform, where this persistence may be unavailable.

Error **kill**(pid: int) 🔗

Kill (terminate) the process identified by the given process ID (`pid`), such as the ID returned by execute in non-blocking mode. See also crash.

Note: This method can also be used to kill processes that were not spawned by the engine.

Note: This method is implemented on Android, iOS, Linux, macOS and Windows.

Error **move_to_trash**(path: String) const 🔗

Moves the file or directory at the given `path` to the system's recycle bin. See also DirAccess.remove.

The method takes only global paths, so you may need to use ProjectSettings.globalize_path. Do not use it for files in `res://` as it will not work in exported projects.

Returns @GlobalScope.FAILED if the file or directory cannot be found, or the system does not support this method.

GDScript

```
var file_to_remove = "user://slot1.save"
OS.move_to_trash(ProjectSettings.globalize_path(file_to_remove)
)
```

C#

```
var fileToRemove = "user://slot1.save";
OS.MoveToTrash(ProjectSettings.GlobalizePath(fileToRemove));
```

Note: This method is implemented on Android, Linux, macOS and Windows.

Note: If the user has disabled the recycle bin on their system, the file will be permanently deleted instead.

---

void **open_midi_inputs**() 🔗

Initializes the singleton for the system MIDI driver, allowing Godot to receive InputEventMIDI. See also get_connected_midi_inputs and close_midi_inputs.

Note: This method is implemented on Linux, macOS and Windows.

String **read_string_from_stdin**()

Reads a user input string from the standard input (usually the terminal). This operation is *blocking*, which causes the window to freeze if read_string_from_stdin is called on the main thread. The thread calling read_string_from_stdin will block until the program receives a line break in standard input (usually by the user pressing `Enter`).

Note: This method is implemented on Linux, macOS and Windows.

Note: On exported Windows builds, run the console wrapper executable to access the terminal. Otherwise, the standard input will not work correctly. If you need a single executable with console support, use a custom build compiled with the `windows_subsystem=console` flag.

---

bool **request_permission**(name: String)

Requests permission from the OS for the given `name`. Returns `true` if the permission has been successfully granted.

Note: This method is currently only implemented on Android, to specifically request permission for `"RECORD_AUDIO"` by `AudioDriverOpenSL`.

---

bool **request_permissions**()

Requests *dangerous* permissions from the OS. Returns `true` if permissions have been successfully granted.

Note: This method is only implemented on Android. Normal permissions are automatically granted at install time in Android applications.

---

void **revoke_granted_permissions**() 🔗

On macOS (sandboxed applications only), this function clears list of user selected folders accessible to the application.

---

void **set_environment**(variable: String, value: String) const 🔗

Sets the value of the environment variable `variable` to `value`. The environment variable will be set for the Godot process and any process executed with execute after running set_environment. The environment variable will *not* persist to processes run after the Godot process was terminated.

Note: Environment variable names are case-sensitive on all platforms except Windows. The `variable` name cannot be empty or include the `=` character. On Windows, there is a 32767 characters limit for the combined length of `variable`, `value`, and the `=` and null terminator characters that will be registered in the environment block.

---

void **set_restart_on_exit**(restart: bool, arguments: PackedStringArray = PackedStringArray()) 🔗

If `restart` is `true`, restarts the project automatically when it is exited with SceneTree.quit or Node.NOTIFICATION_WM_CLOSE_REQUEST. Command-line `arguments` can be supplied. To restart the project with the same command line arguments as originally used to run the project, pass get_cmdline_args as the value for `arguments`.

This method can be used to apply setting changes that require a restart. See also is_restart_on_exit_set and get_restart_on_exit_arguments.

Note: This method is only effective on desktop platforms, and only when the project isn't started from the editor. It will have no effect on mobile and Web platforms, or when the project is started from the editor.

Note: If the project process crashes or is *killed* by the user (by sending `SIGKILL` instead of the usual `SIGTERM`), the project won't restart automatically.

Error **set_thread_name**(name: String)

Assigns the given name to the current thread. Returns @GlobalScope.ERR_UNAVAILABLE if unavailable on the current platform.

---

void **set_use_file_access_save_and_swap**(enabled: bool)

If `enabled` is `true`, when opening a file for writing, a temporary file is used in its place. When closed, it is automatically applied to the target file.

This can useful when files may be opened by other applications, such as antiviruses, text editors, or even the Godot editor itself.

Error **shell_open**(uri: String)

Requests the OS to open a resource identified by `uri` with the most appropriate program. For example:

- `OS.shell_open("C:\\Users\name\Downloads")` on Windows opens the file explorer at the user's Downloads folder.
- `OS.shell_open("https://godotengine.org")` opens the default web browser on the official Godot website.
- `OS.shell_open("mailto:example@example.com")` opens the default email client with the "To" field set to `example@example.com`. See RFC 2368 - The [code]mailto[/code] URL scheme [https://datatracker.ietf.org/doc/html/rfc2368] for a list of fields that can be added.

Use ProjectSettings.globalize_path to convert a `res://` or `user://` project path into a system path for use with this method.

Note: Use String.uri_encode to encode characters within URLs in a URL-safe, portable way. This is especially required for line breaks. Otherwise, shell_open may not work correctly in a project exported to the Web platform.

Note: This method is implemented on Android, iOS, Web, Linux, macOS and Windows.

Error **shell_show_in_file_manager**(file_or_dir_path: String, open_folder: bool = true) 🔗

Requests the OS to open the file manager, navigate to the given `file_or_dir_path` and select the target file or folder.

If `open_folder` is `true` and `file_or_dir_path` is a valid directory path, the OS will open the file manager and navigate to the target folder without selecting anything.

Use ProjectSettings.globalize_path to convert a `res://` or `user://` project path into a system path to use with this method.

Note: This method is currently only implemented on Windows and macOS. On other platforms, it will fallback to shell_open with a directory path of `file_or_dir_path` prefixed with `file://`.

---

void **unset_environment**(variable: String) const 🔗

Removes the given environment variable from the current environment, if it exists. The `variable` name cannot be empty or include the `=` character. The environment variable will be removed for the Godot process and any process executed with execute after running unset_environment. The removal of the environment variable will *not* persist to processes run after the Godot process was terminated.

Note: Environment variable names are case-sensitive on all platforms except Windows.

PackedDataContainerRef

Inherits: RefCounted **<** Object

An internal class used by PackedDataContainer to pack nested arrays and dictionaries.

Description

When packing nested containers using PackedDataContainer, they are recursively packed into **PackedDataContainerRef** (only applies to Array and Dictionary). Their data can be retrieved the same way as from PackedDataContainer.

```
var packed = PackedDataContainer.new()
packed.pack([1, 2, 3, ["abc", "def"], 4, 5, 6])

for element in packed:
    if element is PackedDataContainerRef:
        for subelement in element:
            print("::", subelement)
    else:
        print(element)

# Prints:
# 1
# 2
# 3
# ::abc
# ::def
# 4
# 5
# 6
```

Methods

int size() const

Method Descriptions

int **size**() const 🔗

Returns the size of the packed container (see Array.size and Dictionary.size).

PacketPeer

Inherits: RefCounted **<** Object

Inherited By: ENetPacketPeer, MultiplayerPeer, PacketPeerDTLS, PacketPeerExtension, PacketPeerStream, PacketPeerUDP, WebRTCDataChannel, WebSocketPeer

Abstraction and base class for packet-based protocols.

Description

PacketPeer is an abstraction and base class for packet-based protocols (such as UDP). It provides an API for sending and receiving packets both as raw data or variables. This makes it easy to transfer data over a protocol, without having to encode data as low-level bytes or having to worry about network ordering.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Properties

int	encode_buffer_max_size	`8388608`

Methods

int	get_available_packet_count() const
PackedByteArray	get_packet()
Error	get_packet_error() const
Variant	get_var(allow_objects: bool = false)
Error	put_packet(buffer: PackedByteArray)
Error	put_var(var: Variant, full_objects: bool = false)

Property Descriptions

int **encode_buffer_max_size** = `8388608` 🔗

- void **set_encode_buffer_max_size**(value: int)
- int **get_encode_buffer_max_size**()

Maximum buffer size allowed when encoding Variants. Raise this value to support heavier memory allocations.

The put_var method allocates memory on the stack, and the buffer used will grow automatically to the closest power of two to match the size of the Variant. If the Variant is bigger than encode_buffer_max_size, the method will error out with @GlobalScope.ERR_OUT_OF_MEMORY.

Method Descriptions

int **get_available_packet_count**() const 🔗

Returns the number of packets currently available in the ring-buffer.

PackedByteArray **get_packet**() 🔗

Gets a raw packet.

Error **get_packet_error**() const 🔗

Returns the error state of the last packet received (via get_packet and get_var).

Variant **get_var**(allow_objects: bool = false) 🔗

Gets a Variant. If `allow_objects` is `true`, decoding objects is allowed.

Internally, this uses the same decoding mechanism as the @GlobalScope.bytes_to_var method.

Warning: Deserialized objects can contain code which gets executed. Do not use this option if the serialized object comes from untrusted sources to avoid potential security threats such as remote code execution.

Error **put_packet**(buffer: PackedByteArray) 🔗

Sends a raw packet.

Error **put_var**(var: Variant, full_objects: bool = false) 🔗

Sends a Variant as a packet. If `full_objects` is `true`, encoding objects is allowed (and can potentially include code).

Internally, this uses the same encoding mechanism as the @GlobalScope.var_to_bytes method.

PacketPeerDTLS

Inherits: PacketPeer **<** RefCounted **<** Object

DTLS packet peer.

Description

This class represents a DTLS peer connection. It can be used to connect to a DTLS server, and is returned by DTLSServer.take_connection.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Warning: TLS certificate revocation and certificate pinning are currently not supported. Revoked certificates are accepted as long as they are otherwise valid. If this is a concern, you may want to use automatically managed certificates with a short validity period.

Methods

Error	connect_to_peer(packet_peer: PacketPeerUDP, hostname: String, client_options: TLSOptions = null)
void	disconnect_from_peer()

Status	get_status() const
void	poll()

Enumerations

enum **Status**: 🔗

Status **STATUS_DISCONNECTED** = `0`

A status representing a **PacketPeerDTLS** that is disconnected.

Status **STATUS_HANDSHAKING** = `1`

A status representing a **PacketPeerDTLS** that is currently performing the handshake with a remote peer.

Status **STATUS_CONNECTED** = `2`

A status representing a **PacketPeerDTLS** that is connected to a remote peer.

Status **STATUS_ERROR** = `3`

A status representing a **PacketPeerDTLS** in a generic error state.

Status **STATUS_ERROR_HOSTNAME_MISMATCH** = `4`

An error status that shows a mismatch in the DTLS certificate domain presented by the host and the domain requested for validation.

Method Descriptions

Error **connect_to_peer**(packet_peer: PacketPeerUDP, hostname: String, client_options: TLSOptions = null)

Connects a `packet_peer` beginning the DTLS handshake using the underlying PacketPeerUDP which must be connected (see PacketPeerUDP.connect_to_host). You can optionally specify the `client_options` to be used while verifying the TLS connections. See TLSOptions.client and TLSOptions.client_unsafe.

---

void **disconnect_from_peer**()

Disconnects this peer, terminating the DTLS session.

Status **get_status**() const

Returns the status of the connection. See Status for values.

---

void **poll**()

Poll the connection to check for incoming packets. Call this frequently to update the status and keep the connection working.

PacketPeerExtension

Inherits: PacketPeer **<** RefCounted **<** Object

There is currently no description for this class. Please help us by contributing one!

Methods

int	_get_available_packet_count() virtual const
int	_get_max_packet_size() virtual const
Error	_get_packet(r_buffer: `const uint8_t **`, r_buffer_size: `int32_t*`) virtual
Error	_put_packet(p_buffer: `const uint8_t*`, p_buffer_size: int) virtual

Method Descriptions

int **_get_available_packet_count**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

int **_get_max_packet_size**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Error **_get_packet**(r_buffer: `const uint8_t **`, r_buffer_size: `int32_t*`) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

Error **_put_packet**(p_buffer: `const uint8_t*`, p_buffer_size: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

PacketPeerStream

Inherits: PacketPeer **<** RefCounted **<** Object

Wrapper to use a PacketPeer over a StreamPeer.

Description

PacketStreamPeer provides a wrapper for working using packets over a stream. This allows for using packet based code with StreamPeers. PacketPeerStream implements a custom protocol over the StreamPeer, so the user should not read or write to the wrapped StreamPeer directly.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Properties

int	input_buffer_max_size	`65532`
int	output_buffer_max_size	`65532`
StreamPeer	stream_peer	

Property Descriptions

int **input_buffer_max_size** = `65532` 🔗

- void **set_input_buffer_max_size**(value: int)
- int **get_input_buffer_max_size**()

There is currently no description for this property. Please help us by contributing one!

---

int **output_buffer_max_size** = `65532` 🔗

- void **set_output_buffer_max_size**(value: int)
- int **get_output_buffer_max_size**()

There is currently no description for this property. Please help us by contributing one!

StreamPeer **stream_peer** 🔗

- void **set_stream_peer**(value: StreamPeer)
- StreamPeer **get_stream_peer**()

The wrapped StreamPeer object.

PacketPeerUDP

Inherits: PacketPeer **<** RefCounted **<** Object

UDP packet peer.

Description

UDP packet peer. Can be used to send raw UDP packets as well as Variants.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Methods

Error	bind(port: int, bind_address: String = "*", recv_buf_size: int = 65536)
void	close()
Error	connect_to_host(host: String, port: int)
int	get_local_port() const
String	get_packet_ip() const

int	get_packet_port() const
bool	is_bound() const
bool	is_socket_connected() const
Error	join_multicast_group(multicast_address: String, interface_name: String)
Error	leave_multicast_group(multicast_address: String, interface_name: String)
void	set_broadcast_enabled(enabled: bool)
Error	set_dest_address(host: String, port: int)
Error	wait()

Method Descriptions

Error **bind**(port: int, bind_address: String = "*", recv_buf_size: int = 65536) 🔗

Binds this **PacketPeerUDP** to the specified `port` and `bind_address` with a buffer size `recv_buf_size`, allowing it to receive incoming packets.

If `bind_address` is set to `"*"` (default), the peer will be bound on all available addresses (both IPv4 and IPv6).

If `bind_address` is set to `"0.0.0.0"` (for IPv4) or `"::"` (for IPv6), the peer will be bound to all available addresses matching that IP type.

If `bind_address` is set to any valid address (e.g. `"192.168.1.101"`, `"::1"`, etc.), the peer will only be bound to the interface with that address (or fail if no interface with the given address exists).

---

void **close**() 🔗

Closes the **PacketPeerUDP**'s underlying UDP socket.

Error **connect_to_host**(host: String, port: int) 🔗

Calling this method connects this UDP peer to the given `host`/`port` pair. UDP is in reality connectionless, so this option only means that incoming packets from different addresses are automatically discarded, and that outgoing packets are always sent to the connected address (future calls to set_dest_address are not allowed). This method does not send any data to the remote peer, to do that, use PacketPeer.put_var or PacketPeer.put_packet as usual. See also UDPServer.

Note: Connecting to the remote peer does not help to protect from malicious attacks like IP spoofing, etc. Think about using an encryption technique like TLS or DTLS if you feel like your application is transferring sensitive information.

---

int **get_local_port**() const 🔗

Returns the local port to which this peer is bound.

String **get_packet_ip**() const 🔗

Returns the IP of the remote peer that sent the last packet(that was received with PacketPeer.get_packet or PacketPeer.get_var).

---

int **get_packet_port**() const 🔗

Returns the port of the remote peer that sent the last packet(that was received with PacketPeer.get_packet or PacketPeer.get_var).

---

bool **is_bound**() const 🔗

Returns whether this **PacketPeerUDP** is bound to an address and can receive packets.

---

bool **is_socket_connected**() const 🔗

Returns `true` if the UDP socket is open and has been connected to a remote address. See connect_to_host.

Error **join_multicast_group**(multicast_address: String, interface_name: String) 🔗

Joins the multicast group specified by `multicast_address` using the interface identified by `interface_name`.

You can join the same multicast group with multiple interfaces. Use IP.get_local_interfaces to know which are available.

Note: Some Android devices might require the `CHANGE_WIFI_MULTICAST_STATE` permission for multicast to work.

Error **leave_multicast_group**(multicast_address: String, interface_name: String)

Removes the interface identified by `interface_name` from the multicast group specified by `multicast_address`.

---

void **set_broadcast_enabled**(enabled: bool)

Enable or disable sending of broadcast packets (e.g. `set_dest_address("255.255.255.255", 4343)`. This option is disabled by default.

Note: Some Android devices might require the `CHANGE_WIFI_MULTICAST_STATE` permission and this option to be enabled to receive broadcast packets too.

Error **set_dest_address**(host: String, port: int)

Sets the destination address and port for sending packets and variables. A hostname will be resolved using DNS if needed.

Note: set_broadcast_enabled must be enabled before sending packets to a broadcast address (e.g. `255.255.255.255`).

Error **wait**()

Waits for a packet to arrive on the bound address. See bind.

Note: wait can't be interrupted once it has been called. This can be worked around by allowing the other party to send a specific "death pill" packet like this:

GDScript

```
socket = PacketPeerUDP.new()
# Server
socket.set_dest_address("127.0.0.1", 789)
socket.put_packet("Time to stop".to_ascii_buffer())

# Client
while socket.wait() == OK:
    var data = socket.get_packet().get_string_from_ascii()
    if data == "Time to stop":
        return
```

C#

```
var socket = new PacketPeerUdp();
// Server
socket.SetDestAddress("127.0.0.1", 789);
socket.PutPacket("Time to stop".ToAsciiBuffer());

// Client
while (socket.Wait() == OK)
{
    string data = socket.GetPacket().GetStringFromASCII();
    if (data == "Time to stop")
    {
        return;
    }
}
```

PCKPacker

Inherits: RefCounted < Object

Creates packages that can be loaded into a running project.

Description

The **PCKPacker** is used to create packages that can be loaded into a running project using ProjectSettings.load_resource_pack.

GDScript

```
var packer = PCKPacker.new()
packer.pck_start("test.pck")
packer.add_file("res://text.txt", "text.txt")
packer.flush()
```

C#

```
var packer = new PckPacker();
packer.PckStart("test.pck");
packer.AddFile("res://text.txt", "text.txt");
packer.Flush();
```

The above **PCKPacker** creates package `test.pck`, then adds a file named `text.txt` at the root of the package.

Methods

Error	add_file(pck_path: String, source_path: String, encrypt: bool = false)
Error	flush(verbose: bool = false)
Error	pck_start(pck_name: String, alignment: int = 32, key: String = "0000000000000000000000000000000000000000000000000000000000000000", encrypt_directory: bool = false)

Method Descriptions

Error **add_file**(pck_path: String, source_path: String, encrypt: bool = false)

Adds the `source_path` file to the current PCK package at the `pck_path` internal path (should start with `res://`).

Error **flush**(verbose: bool = false)

Writes the files specified using all add_file calls since the last flush. If `verbose` is `true`, a list of files added will be printed to the console for easier debugging.

Error **pck_start**(pck_name: String, alignment: int = 32, key: String = "0000000000000000000000000000000000000000000000000000000000000000", encrypt_directory: bool = false)

Creates a new PCK file with the name `pck_name`. The `.pck` file extension isn't added automatically, so it should be part of `pck_name` (even though it's not required).

Performance

Inherits: Object

Exposes performance-related data.

Description

This class provides access to a number of different monitors related to performance, such as memory usage, draw calls, and FPS. These are the same as the values displayed in the **Monitor** tab in the editor's **Debugger** panel. By using the get_monitor method of this class, you can access this data from your code.

You can add custom monitors using the add_custom_monitor method. Custom monitors are available in **Monitor** tab in the editor's **Debugger** panel together with built-in monitors.

Note: Some of the built-in monitors are only available in debug mode and will always return `0` when used in a project exported in release mode.

Note: Some of the built-in monitors are not updated in real-time for performance reasons, so there may be a delay of up to 1 second between changes.

Note: Custom monitors do not support negative values. Negative values are clamped to 0.

Methods

void	add_custom_monitor(id: StringName, callable: Callable, arguments: Array = [])
Variant	get_custom_monitor(id: StringName)
Array[StringName]	get_custom_monitor_names()
float	get_monitor(monitor: Monitor) const
int	get_monitor_modification_time()
bool	has_custom_monitor(id: StringName)
void	remove_custom_monitor(id: StringName)

Enumerations

enum **Monitor**: 🔗

Monitor **TIME_FPS** = `0`

The number of frames rendered in the last second. This metric is only updated once per second, even if queried more often. *Higher is better.*

Monitor **TIME_PROCESS** = `1`

Time it took to complete one frame, in seconds. *Lower is better.*

Monitor **TIME_PHYSICS_PROCESS** = `2`

Time it took to complete one physics frame, in seconds. *Lower is better.*

Monitor **TIME_NAVIGATION_PROCESS** = `3`

Time it took to complete one navigation step, in seconds. This includes navigation map updates as well as agent avoidance calculations. *Lower is better.*

Monitor **MEMORY_STATIC** = `4`

Static memory currently used, in bytes. Not available in release builds. *Lower is better.*

Monitor **MEMORY_STATIC_MAX** = `5`

Available static memory. Not available in release builds. *Lower is better.*

Monitor **MEMORY_MESSAGE_BUFFER_MAX** = `6`

Largest amount of memory the message queue buffer has used, in bytes. The message queue is used for deferred functions calls and notifications. *Lower is better.*

Monitor **OBJECT_COUNT** = `7`

Number of objects currently instantiated (including nodes). *Lower is better.*

Monitor **OBJECT_RESOURCE_COUNT** = `8`

Number of resources currently used. *Lower is better.*

Monitor **OBJECT_NODE_COUNT** = `9`

Number of nodes currently instantiated in the scene tree. This also includes the root node. *Lower is better.*

Monitor **OBJECT_ORPHAN_NODE_COUNT** = `10`

Number of orphan nodes, i.e. nodes which are not parented to a node of the scene tree. *Lower is better.*

Monitor **RENDER_TOTAL_OBJECTS_IN_FRAME** = `11`

The total number of objects in the last rendered frame. This metric doesn't include culled objects (either via hiding nodes, frustum culling or occlusion culling). *Lower is better.*

Monitor **RENDER_TOTAL_PRIMITIVES_IN_FRAME** = `12`

The total number of vertices or indices rendered in the last rendered frame. This metric doesn't include primitives from culled objects (either via hiding nodes, frustum culling or occlusion culling). Due to the depth prepass and shadow passes, the number of primitives is always higher than the actual number of vertices in the scene (typically double or triple the original vertex count). *Lower is better.*

Monitor **RENDER_TOTAL_DRAW_CALLS_IN_FRAME** = `13`

The total number of draw calls performed in the last rendered frame. This metric doesn't include culled objects (either via hiding nodes, frustum culling or occlusion culling), since they do not result in draw calls. *Lower is better.*

Monitor **RENDER_VIDEO_MEM_USED** = `14`

The amount of video memory used (texture and vertex memory combined, in bytes). Since this metric also includes miscellaneous allocations, this value is always greater than the sum of RENDER_TEXTURE_MEM_USED and RENDER_BUFFER_MEM_USED. *Lower is better.*

Monitor **RENDER_TEXTURE_MEM_USED** = `15`

The amount of texture memory used (in bytes). *Lower is better.*

Monitor **RENDER_BUFFER_MEM_USED** = `16`

The amount of render buffer memory used (in bytes). *Lower is better.*

Monitor **PHYSICS_2D_ACTIVE_OBJECTS** = `17`

Number of active RigidBody2D nodes in the game. *Lower is better.*

Monitor **PHYSICS_2D_COLLISION_PAIRS** = `18`

Number of collision pairs in the 2D physics engine. *Lower is better.*

Monitor **PHYSICS_2D_ISLAND_COUNT** = `19`

Number of islands in the 2D physics engine. *Lower is better.*

Monitor **PHYSICS_3D_ACTIVE_OBJECTS** = `20`

Number of active RigidBody3D and VehicleBody3D nodes in the game. *Lower is better.*

Monitor **PHYSICS_3D_COLLISION_PAIRS** = `21`

Number of collision pairs in the 3D physics engine. *Lower is better.*

Monitor **PHYSICS_3D_ISLAND_COUNT** = `22`

Number of islands in the 3D physics engine. *Lower is better.*

Monitor **AUDIO_OUTPUT_LATENCY** = `23`

Output latency of the AudioServer. Equivalent to calling AudioServer.get_output_latency, it is not recommended to call this every frame.

Monitor **NAVIGATION_ACTIVE_MAPS** = `24`

Number of active navigation maps in the NavigationServer3D. This also includes the two empty default navigation maps created by World2D and World3D.

Monitor **NAVIGATION_REGION_COUNT** = `25`

Number of active navigation regions in the NavigationServer3D.

Monitor **NAVIGATION_AGENT_COUNT** = `26`

Number of active navigation agents processing avoidance in the NavigationServer3D.

Monitor **NAVIGATION_LINK_COUNT** = `27`

Number of active navigation links in the NavigationServer3D.

Monitor **NAVIGATION_POLYGON_COUNT** = `28`

Number of navigation mesh polygons in the NavigationServer3D.

Monitor **NAVIGATION_EDGE_COUNT** = `29`

Number of navigation mesh polygon edges in the NavigationServer3D.

Monitor **NAVIGATION_EDGE_MERGE_COUNT** = `30`

Number of navigation mesh polygon edges that were merged due to edge key overlap in the NavigationServer3D.

Monitor **NAVIGATION_EDGE_CONNECTION_COUNT** = `31`

Number of polygon edges that are considered connected by edge proximity NavigationServer3D.

Monitor **NAVIGATION_EDGE_FREE_COUNT** = `32`

Number of navigation mesh polygon edges that could not be merged in the NavigationServer3D. The edges still may be connected by edge proximity or with links.

Monitor **MONITOR_MAX** = `33`

Represents the size of the Monitor enum.

Method Descriptions

void **add_custom_monitor**(id: StringName, callable: Callable, arguments: Array = [])

Adds a custom monitor with the name `id`. You can specify the category of the monitor using slash delimiters in `id` (for example: `"Game/NumberOfNPCs"`). If there is more than one slash delimiter, then the default category is used. The default category is `"Custom"`. Prints an error if given `id` is already present.

GDScript

```
func _ready():
    var monitor_value = Callable(self, "get_monitor_value")

    # Adds monitor with name "MyName" to category "MyCategory".
    Performance.add_custom_monitor("MyCategory/MyMonitor",
monitor_value)

    # Adds monitor with name "MyName" to category "Custom".
    # Note: "MyCategory/MyMonitor" and "MyMonitor" have same
name but different IDs, so the code is valid.
    Performance.add_custom_monitor("MyMonitor", monitor_value)
    # Adds monitor with name "MyName" to category "Custom".
    # Note: "MyMonitor" and "Custom/MyMonitor" have same name
and same category but different IDs, so the code is valid.
    Performance.add_custom_monitor("Custom/MyMonitor",
monitor_value)

    # Adds monitor with name
"MyCategoryOne/MyCategoryTwo/MyMonitor" to category "Custom".

Performance.add_custom_monitor("MyCategoryOne/MyCategoryTwo/MyM
onitor", monitor_value)

func get_monitor_value():
    return randi() % 25
```

C#

```
public override void _Ready()
{
    var monitorValue = new Callable(this,
MethodName.GetMonitorValue);

    // Adds monitor with name "MyName" to category
"MyCategory".
    Performance.AddCustomMonitor("MyCategory/MyMonitor",
monitorValue);
    // Adds monitor with name "MyName" to category "Custom".
    // Note: "MyCategory/MyMonitor" and "MyMonitor" have same
name but different ids so the code is valid.
    Performance.AddCustomMonitor("MyMonitor", monitorValue);

    // Adds monitor with name "MyName" to category "Custom".
    // Note: "MyMonitor" and "Custom/MyMonitor" have same name
and same category but different ids so the code is valid.
    Performance.AddCustomMonitor("Custom/MyMonitor",
monitorValue);

    // Adds monitor with name
"MyCategoryOne/MyCategoryTwo/MyMonitor" to category "Custom".

Performance.AddCustomMonitor("MyCategoryOne/MyCategoryTwo/MyMon
itor", monitorValue);
}

public int GetMonitorValue()
{
    return GD.Randi() % 25;
}
```

The debugger calls the callable to get the value of custom monitor. The callable must return a zero or positive integer or floating-point number.

Callables are called with arguments supplied in argument array.

Variant **get_custom_monitor**(id: StringName) 🔗

Returns the value of custom monitor with given `id`. The callable is called to get the value of custom monitor. See also has_custom_monitor. Prints an error if the given `id` is absent.

Array[StringName] **get_custom_monitor_names**() 🔗

Returns the names of active custom monitors in an Array.

---

float **get_monitor**(monitor: Monitor) const 🔗

Returns the value of one of the available built-in monitors. You should provide one of the Monitor constants as the argument, like this:

GDScript

```
print(Performance.get_monitor(Performance.TIME_FPS)) # Prints
the FPS to the console.
```

C#

```
GD.Print(Performance.GetMonitor(Performance.Monitor.TimeFps)); // Prints the FPS to the console.
```

See get_custom_monitor to query custom performance monitors' values.

---

int **get_monitor_modification_time**() 🔗

Returns the last tick in which custom monitor was added/removed (in microseconds since the engine started). This is set to Time.get_ticks_usec when the monitor is updated.

---

bool **has_custom_monitor**(id: StringName) 🔗

Returns `true` if custom monitor with the given `id` is present, `false` otherwise.

---

void **remove_custom_monitor**(id: StringName) 🔗

Removes the custom monitor with given `id`. Prints an error if the given `id` is already absent.

PhysicsDirectBodyState2D

Inherits: Object

Inherited By: PhysicsDirectBodyState2DExtension

Provides direct access to a physics body in the PhysicsServer2D.

Description

Provides direct access to a physics body in the PhysicsServer2D, allowing safe changes to physics properties. This object is passed via the direct state callback of RigidBody2D, and is intended for changing the direct state of that body. See RigidBody2D._integrate_forces.

Tutorials

- Physics introduction
- Ray-casting

Properties

float	angular_velocity
Vector2	center_of_mass
Vector2	center_of_mass_local
float	inverse_inertia
float	inverse_mass
Vector2	linear_velocity

bool	sleeping
float	step
float	total_angular_damp
Vector2	total_gravity
float	total_linear_damp
Transform2D	transform

Methods

void	add_constant_central_force(force: Vector2 = Vector2(0, 0))
void	add_constant_force(force: Vector2, position: Vector2 = Vector2(0, 0))
void	add_constant_torque(torque: float)
void	apply_central_force(force: Vector2 = Vector2(0, 0))
void	apply_central_impulse(impulse: Vector2)
void	apply_force(force: Vector2, position: Vector2 = Vector2(0, 0))

void	apply_impulse(impulse: Vector2, position: Vector2 = Vector2(0, 0))
void	apply_torque(torque: float)
void	apply_torque_impulse(impulse: float)
Vector2	get_constant_force() const
float	get_constant_torque() const
RID	get_contact_collider(contact_idx: int) const
int	get_contact_collider_id(contact_idx: int) const
Object	get_contact_collider_object(contact_idx: int) const
Vector2	get_contact_collider_position(contact_idx: int) const
int	get_contact_collider_shape(contact_idx: int) const
Vector2	get_contact_collider_velocity_at_position(contact_idx: int) const
int	get_contact_count() const
Vector2	get_contact_impulse(contact_idx: int) const
Vector2	get_contact_local_normal(contact_idx: int) const
Vector2	get_contact_local_position(contact_idx: int) const

int	get_contact_local_shape(contact_idx: int) const
Vector2	get_contact_local_velocity_at_position(contact_idx: int) const
PhysicsDirectSpaceState2D	get_space_state()
Vector2	get_velocity_at_local_position(local_position: Vector2) const
void	integrate_forces()
void	set_constant_force(force: Vector2)
void	set_constant_torque(torque: float)

Property Descriptions

float **angular_velocity**

- void **set_angular_velocity**(value: float)
- float **get_angular_velocity**()

The body's rotational velocity in *radians* per second.

Vector2 **center_of_mass**

- Vector2 **get_center_of_mass**()

The body's center of mass position relative to the body's center in the global coordinate system.

Vector2 **center_of_mass_local** 🔗

- Vector2 **get_center_of_mass_local**()

The body's center of mass position in the body's local coordinate system.

---

float **inverse_inertia** 🔗

- float **get_inverse_inertia**()

The inverse of the inertia of the body.

---

float **inverse_mass** 🔗

- float **get_inverse_mass**()

The inverse of the mass of the body.

Vector2 **linear_velocity** 🔗

- void **set_linear_velocity**(value: Vector2)
- Vector2 **get_linear_velocity**()

The body's linear velocity in pixels per second.

---

bool **sleeping** 🔗

- void **set_sleep_state**(value: bool)
- bool **is_sleeping**()

If `true`, this body is currently sleeping (not active).

---

float **step** 🔗

- float **get_step**()

The timestep (delta) used for the simulation.

---

float **total_angular_damp** 🔗

- float **get_total_angular_damp**()

The rate at which the body stops rotating, if there are not any other forces moving it.

Vector2 **total_gravity**

- Vector2 **get_total_gravity**()

The total gravity vector being currently applied to this body.

---

float **total_linear_damp**

- float **get_total_linear_damp**()

The rate at which the body stops moving, if there are not any other forces moving it.

Transform2D **transform**

- void **set_transform**(value: Transform2D)
- Transform2D **get_transform**()

The body's transformation matrix.

Method Descriptions

void **add_constant_central_force**(force: Vector2 = Vector2(0, 0))

Adds a constant directional force without affecting rotation that keeps being applied over time until cleared with `constant_force = Vector2(0, 0)`.

This is equivalent to using add_constant_force at the body's center of mass.

---

void **add_constant_force**(force: Vector2, position: Vector2 = Vector2(0, 0))

Adds a constant positioned force to the body that keeps being applied over time until cleared with `constant_force = Vector2(0, 0)`.

`position` is the offset from the body origin in global coordinates.

---

void **add_constant_torque**(torque: float) 🔗

Adds a constant rotational force without affecting position that keeps being applied over time until cleared with `constant_torque = 0`.

---

void **apply_central_force**(force: Vector2 = Vector2(0, 0)) 🔗

Applies a directional force without affecting rotation. A force is time dependent and meant to be applied every physics update.

This is equivalent to using apply_force at the body's center of mass.

---

void **apply_central_impulse**(impulse: Vector2) 🔗

Applies a directional impulse without affecting rotation.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

This is equivalent to using apply_impulse at the body's center of mass.

---

void **apply_force**(force: Vector2, position: Vector2 = Vector2(0, 0)) 🔗

Applies a positioned force to the body. A force is time dependent and meant to be applied every physics update.

`position` is the offset from the body origin in global coordinates.

---

void **apply_impulse**(impulse: Vector2, position: Vector2 = Vector2(0, 0)) 🔗

Applies a positioned impulse to the body.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

`position` is the offset from the body origin in global coordinates.

void **apply_torque**(torque: float)

Applies a rotational force without affecting position. A force is time dependent and meant to be applied every physics update.

Note: inverse_inertia is required for this to work. To have inverse_inertia, an active CollisionShape2D must be a child of the node, or you can manually set inverse_inertia.

void **apply_torque_impulse**(impulse: float)

Applies a rotational impulse to the body without affecting the position.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

Note: inverse_inertia is required for this to work. To have inverse_inertia, an active CollisionShape2D must be a child of the node, or you can manually set inverse_inertia.

Vector2 **get_constant_force**() const

Returns the body's total constant positional forces applied during each physics update.

See add_constant_force and add_constant_central_force.

float **get_constant_torque**() const

Returns the body's total constant rotational forces applied during each physics update.

See add_constant_torque.

RID **get_contact_collider**(contact_idx: int) const

Returns the collider's RID.

int **get_contact_collider_id**(contact_idx: int) const 🔗

Returns the collider's object id.

Object **get_contact_collider_object**(contact_idx: int) const 🔗

Returns the collider object. This depends on how it was created (will return a scene node if such was used to create it).

Vector2 **get_contact_collider_position**(contact_idx: int) const 🔗

Returns the position of the contact point on the collider in the global coordinate system.

---

int **get_contact_collider_shape**(contact_idx: int) const 🔗

Returns the collider's shape index.

Vector2 **get_contact_collider_velocity_at_position**(contact_idx: int) const 🔗

Returns the velocity vector at the collider's contact point.

---

int **get_contact_count**() const 🔗

Returns the number of contacts this body has with other bodies.

Note: By default, this returns 0 unless bodies are configured to monitor contacts. See RigidBody2D.contact_monitor.

Vector2 **get_contact_impulse**(contact_idx: int) const 🔗

Returns the impulse created by the contact.

Vector2 **get_contact_local_normal**(contact_idx: int) const 🔗

Returns the local normal at the contact point.

Vector2 **get_contact_local_position**(contact_idx: int) const 🔗

Returns the position of the contact point on the body in the global coordinate system.

---

int **get_contact_local_shape**(contact_idx: int) const 🔗

Returns the local shape index of the collision.

Vector2 **get_contact_local_velocity_at_position**(contact_idx: int) const 🔗

Returns the velocity vector at the body's contact point.

PhysicsDirectSpaceState2D **get_space_state**() 🔗

Returns the current state of the space, useful for queries.

Vector2 **get_velocity_at_local_position**(local_position: Vector2) const 🔗

Returns the body's velocity at the given relative position, including both translation and rotation.

---

void **integrate_forces**() 🔗

Updates the body's linear and angular velocity by applying gravity and damping for the equivalent of one physics tick.

---

void **set_constant_force**(force: Vector2) 🔗

Sets the body's total constant positional forces applied during each physics update.

See add_constant_force and add_constant_central_force.

---

void **set_constant_torque**(torque: float) 🔗

Sets the body's total constant rotational forces applied during each physics update.

See add_constant_torque.

PhysicsDirectBodyState2DExtension

Inherits: PhysicsDirectBodyState2D < Object

Provides virtual methods that can be overridden to create custom PhysicsDirectBodyState2D implementations.

Description

This class extends PhysicsDirectBodyState2D by providing additional virtual methods that can be overridden. When these methods are overridden, they will be called instead of the internal methods of the physics server.

Intended for use with GDExtension to create custom implementations of PhysicsDirectBodyState2D.

Methods

void	_add_constant_central_force(force: Vector2) virtual
void	_add_constant_force(force: Vector2, position: Vector2) virtual
void	_add_constant_torque(torque: float) virtual
void	_apply_central_force(force: Vector2) virtual
void	_apply_central_impulse(impulse: Vector2) virtual
void	_apply_force(force: Vector2, position: Vector2) virtual
void	_apply_impulse(impulse: Vector2, position: Vector2) virtual

void	_apply_torque(torque: float) virtual
void	_apply_torque_impulse(impulse: float) virtual
float	_get_angular_velocity() virtual const
Vector2	_get_center_of_mass() virtual const
Vector2	_get_center_of_mass_local() virtual const
Vector2	_get_constant_force() virtual const
float	_get_constant_torque() virtual const
RID	_get_contact_collider(contact_idx: int) virtual const
int	_get_contact_collider_id(contact_idx: int) virtual const
Object	_get_contact_collider_object(contact_idx: int) virtual const
Vector2	_get_contact_collider_position(contact_idx: int) virtual const
int	_get_contact_collider_shape(contact_idx: int) virtual const
Vector2	_get_contact_collider_velocity_at_position(contact_idx: int) virtual const
int	_get_contact_count() virtual const

Vector2	_get_contact_impulse(contact_idx: int) virtual const
Vector2	_get_contact_local_normal(contact_idx: int) virtual const
Vector2	_get_contact_local_position(contact_idx: int) virtual const
int	_get_contact_local_shape(contact_idx: int) virtual const
Vector2	_get_contact_local_velocity_at_position(contact_idx: int) virtual const
float	_get_inverse_inertia() virtual const
float	_get_inverse_mass() virtual const
Vector2	_get_linear_velocity() virtual const
PhysicsDirectSpaceState2D	_get_space_state() virtual
float	_get_step() virtual const
float	_get_total_angular_damp() virtual const
Vector2	_get_total_gravity() virtual const
float	_get_total_linear_damp() virtual const
Transform2D	_get_transform() virtual const

Vector2	_get_velocity_at_local_position(local_position: Vector2) virtual const
void	_integrate_forces() virtual
bool	_is_sleeping() virtual const
void	_set_angular_velocity(velocity: float) virtual
void	_set_constant_force(force: Vector2) virtual
void	_set_constant_torque(torque: float) virtual
void	_set_linear_velocity(velocity: Vector2) virtual
void	_set_sleep_state(enabled: bool) virtual
void	_set_transform(transform: Transform2D) virtual

Method Descriptions

void **_add_constant_central_force**(force: Vector2) virtual

Overridable version of PhysicsDirectBodyState2D.add_constant_central_force.

void **_add_constant_force**(force: Vector2, position: Vector2) virtual

Overridable version of PhysicsDirectBodyState2D.add_constant_force.

void **_add_constant_torque**(torque: float) virtual

Overridable version of PhysicsDirectBodyState2D.add_constant_torque.

void **_apply_central_force**(force: Vector2) virtual

Overridable version of PhysicsDirectBodyState2D.apply_central_force.

void **_apply_central_impulse**(impulse: Vector2) virtual

Overridable version of PhysicsDirectBodyState2D.apply_central_impulse.

void **_apply_force**(force: Vector2, position: Vector2) virtual

Overridable version of PhysicsDirectBodyState2D.apply_force.

void **_apply_impulse**(impulse: Vector2, position: Vector2) virtual

Overridable version of PhysicsDirectBodyState2D.apply_impulse.

void **_apply_torque**(torque: float) virtual

Overridable version of PhysicsDirectBodyState2D.apply_torque.

void **_apply_torque_impulse**(impulse: float) virtual

Overridable version of PhysicsDirectBodyState2D.apply_torque_impulse.

float **_get_angular_velocity**() virtual const

Implement to override the behavior of PhysicsDirectBodyState2D.angular_velocity and its respective getter.

Vector2 **_get_center_of_mass**() virtual const

Implement to override the behavior of PhysicsDirectBodyState2D.center_of_mass and its respective getter.

Vector2 **_get_center_of_mass_local**() virtual const

Implement to override the behavior of PhysicsDirectBodyState2D.center_of_mass_local and its respective getter.

Vector2 **_get_constant_force**() virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_constant_force.

---

float **_get_constant_torque**() virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_constant_torque.

RID **_get_contact_collider**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_collider.

---

int **_get_contact_collider_id**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_collider_id.

Object **_get_contact_collider_object**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_collider_object.

Vector2 **_get_contact_collider_position**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_collider_position.

---

int **_get_contact_collider_shape**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_collider_shape.

Vector2 **_get_contact_collider_velocity_at_position**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_collider_velocity_at_position.

---

int **_get_contact_count**() virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_count.

Vector2 **_get_contact_impulse**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_impulse.

Vector2 **_get_contact_local_normal**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_local_normal.

Vector2 **_get_contact_local_position**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_local_position.

---

int **_get_contact_local_shape**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_local_shape.

Vector2 **_get_contact_local_velocity_at_position**(contact_idx: int) virtual const 🔗

Overridable version of PhysicsDirectBodyState2D.get_contact_local_velocity_at_position.

---

float **_get_inverse_inertia**() virtual const 🔗

Implement to override the behavior of PhysicsDirectBodyState2D.inverse_inertia and its respective getter.

---

float **_get_inverse_mass**() virtual const 🔗

Implement to override the behavior of PhysicsDirectBodyState2D.inverse_mass and its respective getter.

Vector2 **_get_linear_velocity**() virtual const 🔗

Implement to override the behavior of PhysicsDirectBodyState2D.linear_velocity and its respective getter.

PhysicsDirectSpaceState2D **_get_space_state**() virtual

Overridable version of PhysicsDirectBodyState2D.get_space_state.

---

float **_get_step**() virtual const

Implement to override the behavior of PhysicsDirectBodyState2D.step and its respective getter.

---

float **_get_total_angular_damp**() virtual const

Implement to override the behavior of PhysicsDirectBodyState2D.total_angular_damp and its respective getter.

Vector2 **_get_total_gravity**() virtual const

Implement to override the behavior of PhysicsDirectBodyState2D.total_gravity and its respective getter.

---

float **_get_total_linear_damp**() virtual const

Implement to override the behavior of PhysicsDirectBodyState2D.total_linear_damp and its respective getter.

Transform2D **_get_transform**() virtual const

Implement to override the behavior of PhysicsDirectBodyState2D.transform and its respective getter.

Vector2 **_get_velocity_at_local_position**(local_position: Vector2) virtual const

Overridable version of PhysicsDirectBodyState2D.get_velocity_at_local_position.

---

void **_integrate_forces**() virtual

Overridable version of PhysicsDirectBodyState2D.integrate_forces.

---

bool **_is_sleeping**() virtual const

Implement to override the behavior of PhysicsDirectBodyState2D.sleeping and its respective getter.

---

void **_set_angular_velocity**(velocity: float) virtual

Implement to override the behavior of PhysicsDirectBodyState2D.angular_velocity and its respective setter.

---

void **_set_constant_force**(force: Vector2) virtual

Overridable version of PhysicsDirectBodyState2D.set_constant_force.

---

void **_set_constant_torque**(torque: float) virtual

Overridable version of PhysicsDirectBodyState2D.set_constant_torque.

---

void **_set_linear_velocity**(velocity: Vector2) virtual

Implement to override the behavior of PhysicsDirectBodyState2D.linear_velocity and its respective setter.

---

void **_set_sleep_state**(enabled: bool) virtual

Implement to override the behavior of PhysicsDirectBodyState2D.sleeping and its respective setter.

---

void **_set_transform**(transform: Transform2D) virtual

Implement to override the behavior of PhysicsDirectBodyState2D.transform and its respective setter.

PhysicsDirectBodyState3D

Inherits: Object

Inherited By: PhysicsDirectBodyState3DExtension

Provides direct access to a physics body in the PhysicsServer3D.

Description

Provides direct access to a physics body in the PhysicsServer3D, allowing safe changes to physics properties. This object is passed via the direct state callback of RigidBody3D, and is intended for changing the direct state of that body. See RigidBody3D._integrate_forces.

Tutorials

- Physics introduction
- Ray-casting

Properties

Vector3	angular_velocity
Vector3	center_of_mass
Vector3	center_of_mass_local
Vector3	inverse_inertia
Basis	inverse_inertia_tensor
float	inverse_mass

Vector3	linear_velocity
Basis	principal_inertia_axes
bool	sleeping
float	step
float	total_angular_damp
Vector3	total_gravity
float	total_linear_damp
Transform3D	transform

Methods

void	add_constant_central_force(force: Vector3 = Vector3(0, 0, 0))
void	add_constant_force(force: Vector3, position: Vector3 = Vector3(0, 0, 0))
void	add_constant_torque(torque: Vector3)
void	apply_central_force(force: Vector3 = Vector3(0, 0, 0))

void	apply_central_impulse(impulse: Vector3 = Vector3(0, 0, 0))
void	apply_force(force: Vector3, position: Vector3 = Vector3(0, 0, 0))
void	apply_impulse(impulse: Vector3, position: Vector3 = Vector3(0, 0, 0))
void	apply_torque(torque: Vector3)
void	apply_torque_impulse(impulse: Vector3)
Vector3	get_constant_force() const
Vector3	get_constant_torque() const
RID	get_contact_collider(contact_idx: int) const
int	get_contact_collider_id(contact_idx: int) const
Object	get_contact_collider_object(contact_idx: int) const
Vector3	get_contact_collider_position(contact_idx: int) const
int	get_contact_collider_shape(contact_idx: int) const
Vector3	get_contact_collider_velocity_at_position(contact_idx: int) const
int	get_contact_count() const

Vector3	get_contact_impulse(contact_idx: int) const
Vector3	get_contact_local_normal(contact_idx: int) const
Vector3	get_contact_local_position(contact_idx: int) const
int	get_contact_local_shape(contact_idx: int) const
Vector3	get_contact_local_velocity_at_position(contact_idx: int) const
PhysicsDirectSpaceState3D	get_space_state()
Vector3	get_velocity_at_local_position(local_position: Vector3) const
void	integrate_forces()
void	set_constant_force(force: Vector3)
void	set_constant_torque(torque: Vector3)

Property Descriptions

Vector3 **angular_velocity** 🔗

- void **set_angular_velocity**(value: Vector3)
- Vector3 **get_angular_velocity**()

The body's rotational velocity in *radians* per second.

Vector3 **center_of_mass** 🔗

- Vector3 **get_center_of_mass**()

The body's center of mass position relative to the body's center in the global coordinate system.

Vector3 **center_of_mass_local** 🔗

- Vector3 **get_center_of_mass_local**()

The body's center of mass position in the body's local coordinate system.

Vector3 **inverse_inertia** 🔗

- Vector3 **get_inverse_inertia**()

The inverse of the inertia of the body.

Basis **inverse_inertia_tensor** 🔗

- Basis **get_inverse_inertia_tensor**()

The inverse of the inertia tensor of the body.

---

float **inverse_mass** 🔗

- float **get_inverse_mass**()

The inverse of the mass of the body.

Vector3 **linear_velocity** 🔗

- void **set_linear_velocity**(value: Vector3)
- Vector3 **get_linear_velocity**()

The body's linear velocity in units per second.

Basis **principal_inertia_axes** 🔗

- Basis **get_principal_inertia_axes**()

There is currently no description for this property. Please help us by contributing one!

---

bool **sleeping**

- void **set_sleep_state**(value: bool)
- bool **is_sleeping**()

If `true`, this body is currently sleeping (not active).

---

float **step**

- float **get_step**()

The timestep (delta) used for the simulation.

---

float **total_angular_damp**

- float **get_total_angular_damp**()

The rate at which the body stops rotating, if there are not any other forces moving it.

Vector3 **total_gravity**

- Vector3 **get_total_gravity**()

The total gravity vector being currently applied to this body.

---

float **total_linear_damp**

- float **get_total_linear_damp**()

The rate at which the body stops moving, if there are not any other forces moving it.

Transform3D **transform**

- void **set_transform**(value: Transform3D)
- Transform3D **get_transform**()

The body's transformation matrix.

Method Descriptions

void **add_constant_central_force**(force: Vector3 = Vector3(0, 0, 0))

Adds a constant directional force without affecting rotation that keeps being applied over time until cleared with `constant_force = Vector3(0, 0, 0)`.

This is equivalent to using add_constant_force at the body's center of mass.

void **add_constant_force**(force: Vector3, position: Vector3 = Vector3(0, 0, 0))

Adds a constant positioned force to the body that keeps being applied over time until cleared with `constant_force = Vector3(0, 0, 0)`.

`position` is the offset from the body origin in global coordinates.

void **add_constant_torque**(torque: Vector3)

Adds a constant rotational force without affecting position that keeps being applied over time until cleared with `constant_torque = Vector3(0, 0, 0)`.

void **apply_central_force**(force: Vector3 = Vector3(0, 0, 0))

Applies a directional force without affecting rotation. A force is time dependent and meant to be applied every physics update.

This is equivalent to using apply_force at the body's center of mass.

void **apply_central_impulse**(impulse: Vector3 = Vector3(0, 0, 0))

Applies a directional impulse without affecting rotation.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

This is equivalent to using apply_impulse at the body's center of mass.

void **apply_force**(force: Vector3, position: Vector3 = Vector3(0, 0, 0)) 🔗

Applies a positioned force to the body. A force is time dependent and meant to be applied every physics update.

`position` is the offset from the body origin in global coordinates.

void **apply_impulse**(impulse: Vector3, position: Vector3 = Vector3(0, 0, 0)) 🔗

Applies a positioned impulse to the body.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

`position` is the offset from the body origin in global coordinates.

void **apply_torque**(torque: Vector3) 🔗

Applies a rotational force without affecting position. A force is time dependent and meant to be applied every physics update.

Note: inverse_inertia is required for this to work. To have inverse_inertia, an active CollisionShape3D must be a child of the node, or you can manually set inverse_inertia.

void **apply_torque_impulse**(impulse: Vector3) 🔗

Applies a rotational impulse to the body without affecting the position.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

Note: inverse_inertia is required for this to work. To have inverse_inertia, an active CollisionShape3D must be a child of the node, or you can manually set inverse_inertia.

Vector3 **get_constant_force**() const

Returns the body's total constant positional forces applied during each physics update.

See add_constant_force and add_constant_central_force.

Vector3 **get_constant_torque**() const

Returns the body's total constant rotational forces applied during each physics update.

See add_constant_torque.

RID **get_contact_collider**(contact_idx: int) const

Returns the collider's RID.

---

int **get_contact_collider_id**(contact_idx: int) const

Returns the collider's object id.

Object **get_contact_collider_object**(contact_idx: int) const

Returns the collider object.

Vector3 **get_contact_collider_position**(contact_idx: int) const

Returns the position of the contact point on the collider in the global coordinate system.

---

int **get_contact_collider_shape**(contact_idx: int) const

Returns the collider's shape index.

Vector3 **get_contact_collider_velocity_at_position**(contact_idx: int) const

Returns the linear velocity vector at the collider's contact point.

---

int **get_contact_count**() const

Returns the number of contacts this body has with other bodies.

Note: By default, this returns 0 unless bodies are configured to monitor contacts. See RigidBody3D.contact_monitor.

Vector3 **get_contact_impulse**(contact_idx: int) const

Impulse created by the contact.

Vector3 **get_contact_local_normal**(contact_idx: int) const

Returns the local normal at the contact point.

Vector3 **get_contact_local_position**(contact_idx: int) const

Returns the position of the contact point on the body in the global coordinate system.

---

int **get_contact_local_shape**(contact_idx: int) const

Returns the local shape index of the collision.

Vector3 **get_contact_local_velocity_at_position**(contact_idx: int) const

Returns the linear velocity vector at the body's contact point.

PhysicsDirectSpaceState3D **get_space_state**()

Returns the current state of the space, useful for queries.

Vector3 **get_velocity_at_local_position**(local_position: Vector3) const

Returns the body's velocity at the given relative position, including both translation and rotation.

---

void **integrate_forces**()

Updates the body's linear and angular velocity by applying gravity and damping for the equivalent of one physics tick.

---

void **set_constant_force**(force: Vector3)

Sets the body's total constant positional forces applied during each physics update.

See add_constant_force and add_constant_central_force.

---

void **set_constant_torque**(torque: Vector3)

Sets the body's total constant rotational forces applied during each physics update.

See add_constant_torque.

PhysicsDirectBodyState3DExtension

Inherits: PhysicsDirectBodyState3D **<** Object

Provides virtual methods that can be overridden to create custom PhysicsDirectBodyState3D implementations.

Description

This class extends PhysicsDirectBodyState3D by providing additional virtual methods that can be overridden. When these methods are overridden, they will be called instead of the internal methods of the physics server.

Intended for use with GDExtension to create custom implementations of PhysicsDirectBodyState3D.

Methods

void	_add_constant_central_force(force: Vector3) virtual
void	_add_constant_force(force: Vector3, position: Vector3) virtual
void	_add_constant_torque(torque: Vector3) virtual
void	_apply_central_force(force: Vector3) virtual
void	_apply_central_impulse(impulse: Vector3) virtual
void	_apply_force(force: Vector3, position: Vector3) virtual
void	_apply_impulse(impulse: Vector3, position: Vector3) virtual

void	_apply_torque(torque: Vector3) virtual
void	_apply_torque_impulse(impulse: Vector3) virtual
Vector3	_get_angular_velocity() virtual const
Vector3	_get_center_of_mass() virtual const
Vector3	_get_center_of_mass_local() virtual const
Vector3	_get_constant_force() virtual const
Vector3	_get_constant_torque() virtual const
RID	_get_contact_collider(contact_idx: int) virtual const
int	_get_contact_collider_id(contact_idx: int) virtual const
Object	_get_contact_collider_object(contact_idx: int) virtual const
Vector3	_get_contact_collider_position(contact_idx: int) virtual const
int	_get_contact_collider_shape(contact_idx: int) virtual const
Vector3	_get_contact_collider_velocity_at_position(contact_idx: int) virtual const
int	_get_contact_count() virtual const

Vector3	_get_contact_impulse(contact_idx: int) virtual const
Vector3	_get_contact_local_normal(contact_idx: int) virtual const
Vector3	_get_contact_local_position(contact_idx: int) virtual const
int	_get_contact_local_shape(contact_idx: int) virtual const
Vector3	_get_contact_local_velocity_at_position(contact_idx: int) virtual const
Vector3	_get_inverse_inertia() virtual const
Basis	_get_inverse_inertia_tensor() virtual const
float	_get_inverse_mass() virtual const
Vector3	_get_linear_velocity() virtual const
Basis	_get_principal_inertia_axes() virtual const
PhysicsDirectSpaceState3D	_get_space_state() virtual
float	_get_step() virtual const
float	_get_total_angular_damp() virtual const
Vector3	_get_total_gravity() virtual const

float	_get_total_linear_damp() virtual const
Transform3D	_get_transform() virtual const
Vector3	_get_velocity_at_local_position(local_position: Vector3) virtual const
void	_integrate_forces() virtual
bool	_is_sleeping() virtual const
void	_set_angular_velocity(velocity: Vector3) virtual
void	_set_constant_force(force: Vector3) virtual
void	_set_constant_torque(torque: Vector3) virtual
void	_set_linear_velocity(velocity: Vector3) virtual
void	_set_sleep_state(enabled: bool) virtual
void	_set_transform(transform: Transform3D) virtual

Method Descriptions

void **_add_constant_central_force**(force: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_add_constant_force**(force: Vector3, position: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_add_constant_torque**(torque: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_apply_central_force**(force: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_apply_central_impulse**(impulse: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_apply_force**(force: Vector3, position: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_apply_impulse**(impulse: Vector3, position: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_apply_torque**(torque: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_apply_torque_impulse**(impulse: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_angular_velocity**() virtual const

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_center_of_mass**() virtual const

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_center_of_mass_local**() virtual const

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_constant_force**() virtual const

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_constant_torque**() virtual const

There is currently no description for this method. Please help us by contributing one!

RID **_get_contact_collider**(contact_idx: int) virtual const

There is currently no description for this method. Please help us by contributing one!

---

int **_get_contact_collider_id**(contact_idx: int) virtual const

There is currently no description for this method. Please help us by contributing one!

Object **_get_contact_collider_object**(contact_idx: int) virtual const

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_contact_collider_position**(contact_idx: int) virtual const

There is currently no description for this method. Please help us by contributing one!

---

int **_get_contact_collider_shape**(contact_idx: int) virtual const

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_contact_collider_velocity_at_position**(contact_idx: int) virtual const

There is currently no description for this method. Please help us by contributing one!

---

int **_get_contact_count**() virtual const

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_contact_impulse**(contact_idx: int) virtual const

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_contact_local_normal**(contact_idx: int) virtual const

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_contact_local_position**(contact_idx: int) virtual const

There is currently no description for this method. Please help us by contributing one!

int **_get_contact_local_shape**(contact_idx: int) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_contact_local_velocity_at_position**(contact_idx: int) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_inverse_inertia**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Basis **_get_inverse_inertia_tensor**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

float **_get_inverse_mass**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_linear_velocity**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Basis **_get_principal_inertia_axes**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

PhysicsDirectSpaceState3D **_get_space_state**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_get_step**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_get_total_angular_damp**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_total_gravity**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_get_total_linear_damp**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Transform3D **_get_transform**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_velocity_at_local_position**(local_position: Vector3) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_integrate_forces**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_is_sleeping**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_set_angular_velocity**(velocity: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_set_constant_force**(force: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_set_constant_torque**(torque: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_set_linear_velocity**(velocity: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_set_sleep_state**(enabled: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_set_transform**(transform: Transform3D) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

PhysicsDirectSpaceState2D

Inherits: Object

Inherited By: PhysicsDirectSpaceState2DExtension

Provides direct access to a physics space in the PhysicsServer2D.

Description

Provides direct access to a physics space in the PhysicsServer2D. It's used mainly to do queries against objects and areas residing in a given space.

Tutorials

- Physics introduction
- Ray-casting

Methods

PackedFloat32Array	cast_motion(parameters: PhysicsShapeQueryParameters2D)
Array[Vector2]	collide_shape(parameters: PhysicsShapeQueryParameters2D, max_results: int = 32)

Dictionary	get_rest_info(parameters: PhysicsShapeQueryParameters2D)
Array[Dictionary]	intersect_point(parameters: PhysicsPointQueryParameters2D, max_results: int = 32)
Dictionary	intersect_ray(parameters: PhysicsRayQueryParameters2D)
Array[Dictionary]	intersect_shape(parameters: PhysicsShapeQueryParameters2D, max_results: int = 32)

Method Descriptions

PackedFloat32Array **cast_motion**(parameters: PhysicsShapeQueryParameters2D) 🔗

Checks how far a Shape2D can move without colliding. All the parameters for the query, including the shape and the motion, are supplied through a PhysicsShapeQueryParameters2D object.

Returns an array with the safe and unsafe proportions (between 0 and 1) of the motion. The safe proportion is the maximum fraction of the motion that can be made without a collision. The unsafe proportion is the minimum fraction of the distance that must be moved for a collision. If no collision is detected a result of `[1.0, 1.0]` will be returned.

Note: Any Shape2Ds that the shape is already colliding with e.g. inside of, will be ignored. Use collide_shape to determine the Shape2Ds that the shape is already colliding with.

Array[Vector2] **collide_shape**(parameters: PhysicsShapeQueryParameters2D, max_results: int = 32) 🔗

Checks the intersections of a shape, given through a PhysicsShapeQueryParameters2D object, against the space. The resulting array contains a list of points where the shape intersects another. Like with intersect_shape, the number of returned results can be limited to save processing time.

Returned points are a list of pairs of contact points. For each pair the first one is in the shape passed in PhysicsShapeQueryParameters2D object, second one is in the collided shape from the physics space.

Dictionary **get_rest_info**(parameters: PhysicsShapeQueryParameters2D) 🔗

Checks the intersections of a shape, given through a PhysicsShapeQueryParameters2D object, against the space. If it collides with more than one shape, the nearest one is selected. If the shape did not intersect anything, then an empty dictionary is returned instead.

Note: This method does not take into account the `motion` property of the object. The returned object is a dictionary containing the following fields:

`collider_id`: The colliding object's ID.

`linear_velocity`: The colliding object's velocity Vector2. If the object is an Area2D, the result is `(0, 0)`.

`normal`: The object's surface normal at the intersection point.

`point`: The intersection point.

`rid`: The intersecting object's RID.

`shape`: The shape index of the colliding shape.

Array[Dictionary] **intersect_point**(parameters: PhysicsPointQueryParameters2D, max_results: int = 32)

Checks whether a point is inside any solid shape. Position and other parameters are defined through PhysicsPointQueryParameters2D. The shapes the point is inside of are returned in an array containing dictionaries with the following fields:

`collider`: The colliding object.

`collider_id`: The colliding object's ID.

`rid`: The intersecting object's RID.

`shape`: The shape index of the colliding shape.

The number of intersections can be limited with the `max_results` parameter, to reduce the processing time.

Note: ConcavePolygonShape2Ds and CollisionPolygon2Ds in `Segments` build mode are not solid shapes. Therefore, they will not be detected.

Dictionary **intersect_ray**(parameters: PhysicsRayQueryParameters2D)

Intersects a ray in a given space. Ray position and other parameters are defined through PhysicsRayQueryParameters2D. The returned object is a dictionary with the following fields:

`collider`: The colliding object.

`collider_id`: The colliding object's ID.

`normal`: The object's surface normal at the intersection point, or `Vector2(0, 0)` if the ray starts inside the shape and PhysicsRayQueryParameters2D.hit_from_inside is `true`.

`position`: The intersection point.

`rid`: The intersecting object's RID.

`shape`: The shape index of the colliding shape.

If the ray did not intersect anything, then an empty dictionary is returned instead.

Array[Dictionary] **intersect_shape**(parameters: PhysicsShapeQueryParameters2D, max_results: int = 32) 🔗

Checks the intersections of a shape, given through a PhysicsShapeQueryParameters2D object, against the space. The intersected shapes are returned in an array containing dictionaries with the following fields:

`collider`: The colliding object.

`collider_id`: The colliding object's ID.

`rid`: The intersecting object's RID.

`shape`: The shape index of the colliding shape.

The number of intersections can be limited with the `max_results` parameter, to reduce the processing time.

PhysicsDirectSpaceState2DExtension

Inherits: PhysicsDirectSpaceState2D **<** Object

Provides virtual methods that can be overridden to create custom PhysicsDirectSpaceState2D implementations.

Description

This class extends PhysicsDirectSpaceState2D by providing additional virtual methods that can be overridden. When these methods are overridden, they will be called instead of the internal methods of the physics server.

Intended for use with GDExtension to create custom implementations of PhysicsDirectSpaceState2D.

Methods

bool	_cast_motion(shape_rid: RID, transform: Transform2D, motion: Vector2, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, closest_safe: `float*`, closest_unsafe: `float*`) virtual
bool	_collide_shape(shape_rid: RID, transform: Transform2D, motion: Vector2, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, results: `void*`, max_results: int, result_count: `int32_t*`) virtual

int	_intersect_point(position: Vector2, canvas_instance_id: int, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, results: `PhysicsServer2DExtensionShapeResult*`, max_results: int) virtual
bool	_intersect_ray(from: Vector2, to: Vector2, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, hit_from_inside: bool, result: `PhysicsServer2DExtensionRayResult*`) virtual
int	_intersect_shape(shape_rid: RID, transform: Transform2D, motion: Vector2, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, result: `PhysicsServer2DExtensionShapeResult*`, max_results: int) virtual
bool	_rest_info(shape_rid: RID, transform: Transform2D, motion: Vector2, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, rest_info: `PhysicsServer2DExtensionShapeRestInfo*`) virtual
bool	is_body_excluded_from_query(body: RID) const

Method Descriptions

bool **_cast_motion**(shape_rid: RID, transform: Transform2D, motion: Vector2, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, closest_safe: `float*`, closest_unsafe: `float*`) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_collide_shape**(shape_rid: RID, transform: Transform2D, motion: Vector2, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, results: `void*`, max_results: int, result_count: `int32_t*`) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_intersect_point**(position: Vector2, canvas_instance_id: int, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, results: `PhysicsServer2DExtensionShapeResult*`, max_results: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_intersect_ray**(from: Vector2, to: Vector2, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, hit_from_inside: bool, result: `PhysicsServer2DExtensionRayResult*`) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_intersect_shape**(shape_rid: RID, transform: Transform2D, motion: Vector2, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, result: `PhysicsServer2DExtensionShapeResult*`, max_results: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_rest_info**(shape_rid: RID, transform: Transform2D, motion: Vector2, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, rest_info: `PhysicsServer2DExtensionShapeRestInfo*`) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **is_body_excluded_from_query**(body: RID) const 🔗

There is currently no description for this method. Please help us by contributing one!

PhysicsDirectSpaceState3D

Inherits: Object

Inherited By: PhysicsDirectSpaceState3DExtension

Provides direct access to a physics space in the PhysicsServer3D.

Description

Provides direct access to a physics space in the PhysicsServer3D. It's used mainly to do queries against objects and areas residing in a given space.

Tutorials

- Physics introduction
- Ray-casting

Methods

PackedFloat32Array	cast_motion(parameters: PhysicsShapeQueryParameters3D)
Array[Vector3]	collide_shape(parameters: PhysicsShapeQueryParameters3D, max_results: int = 32)

Dictionary	get_rest_info(parameters: PhysicsShapeQueryParameters3D)
Array[Dictionary]	intersect_point(parameters: PhysicsPointQueryParameters3D, max_results: int = 32)
Dictionary	intersect_ray(parameters: PhysicsRayQueryParameters3D)
Array[Dictionary]	intersect_shape(parameters: PhysicsShapeQueryParameters3D, max_results: int = 32)

Method Descriptions

PackedFloat32Array **cast_motion**(parameters: PhysicsShapeQueryParameters3D) 🔗

Checks how far a Shape3D can move without colliding. All the parameters for the query, including the shape, are supplied through a PhysicsShapeQueryParameters3D object.

Returns an array with the safe and unsafe proportions (between 0 and 1) of the motion. The safe proportion is the maximum fraction of the motion that can be made without a collision. The unsafe proportion is the minimum fraction of the distance that must be moved for a collision. If no collision is detected a result of `[1.0, 1.0]` will be returned.

Note: Any Shape3Ds that the shape is already colliding with e.g. inside of, will be ignored. Use collide_shape to determine the Shape3Ds that the shape is already colliding with.

Array[Vector3] **collide_shape**(parameters: PhysicsShapeQueryParameters3D, max_results: int = 32)

Checks the intersections of a shape, given through a PhysicsShapeQueryParameters3D object, against the space. The resulting array contains a list of points where the shape intersects another. Like with intersect_shape, the number of returned results can be limited to save processing time.

Returned points are a list of pairs of contact points. For each pair the first one is in the shape passed in PhysicsShapeQueryParameters3D object, second one is in the collided shape from the physics space.

Note: This method does not take into account the `motion` property of the object.

Dictionary **get_rest_info**(parameters: PhysicsShapeQueryParameters3D)

Checks the intersections of a shape, given through a PhysicsShapeQueryParameters3D object, against the space. If it collides with more than one shape, the nearest one is selected. The returned object is a dictionary containing the following fields:

`collider_id`: The colliding object's ID.

`linear_velocity`: The colliding object's velocity Vector3. If the object is an Area3D, the result is `(0, 0, 0)`.

`normal`: The object's surface normal at the intersection point.

`point`: The intersection point.

`rid`: The intersecting object's RID.

`shape`: The shape index of the colliding shape.

If the shape did not intersect anything, then an empty dictionary is returned instead.

Note: This method does not take into account the `motion` property of the object.

Array[Dictionary] **intersect_point**(parameters: PhysicsPointQueryParameters3D, max_results: int = 32) 🔗

Checks whether a point is inside any solid shape. Position and other parameters are defined through PhysicsPointQueryParameters3D. The shapes the point is inside of are returned in an array containing dictionaries with the following fields:

`collider`: The colliding object.

`collider_id`: The colliding object's ID.

`rid`: The intersecting object's RID.

`shape`: The shape index of the colliding shape.

The number of intersections can be limited with the `max_results` parameter, to reduce the processing time.

Dictionary **intersect_ray**(parameters: PhysicsRayQueryParameters3D) 🔗

Intersects a ray in a given space. Ray position and other parameters are defined through PhysicsRayQueryParameters3D. The returned object is a dictionary with the following fields:

`collider`: The colliding object.

`collider_id`: The colliding object's ID.

`normal`: The object's surface normal at the intersection point, or `Vector3(0, 0, 0)` if the ray starts inside the shape and PhysicsRayQueryParameters3D.hit_from_inside is `true`.

`position`: The intersection point.

`face_index`: The face index at the intersection point.

Note: Returns a valid number only if the intersected shape is a ConcavePolygonShape3D. Otherwise, `-1` is returned.

`rid`: The intersecting object's RID.

`shape`: The shape index of the colliding shape.

If the ray did not intersect anything, then an empty dictionary is returned instead.

Array[Dictionary] **intersect_shape**(parameters: PhysicsShapeQueryParameters3D, max_results: int = 32) 🔗

Checks the intersections of a shape, given through a PhysicsShapeQueryParameters3D object, against the space. The intersected shapes are returned in an array containing dictionaries with the following fields:

`collider`: The colliding object.

`collider_id`: The colliding object's ID.

`rid`: The intersecting object's RID.

`shape`: The shape index of the colliding shape.

The number of intersections can be limited with the `max_results` parameter, to reduce the processing time.

Note: This method does not take into account the `motion` property of the object.

PhysicsDirectSpaceState3DExtension

Inherits: PhysicsDirectSpaceState3D < Object

Provides virtual methods that can be overridden to create custom PhysicsDirectSpaceState3D implementations.

Description

This class extends PhysicsDirectSpaceState3D by providing additional virtual methods that can be overridden. When these methods are overridden, they will be called instead of the internal methods of the physics server.

Intended for use with GDExtension to create custom implementations of PhysicsDirectSpaceState3D.

Methods

bool	_cast_motion(shape_rid: RID, transform: Transform3D, motion: Vector3, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, closest_safe: `float*`, closest_unsafe: `float*`, info: `PhysicsServer3DExtensionShapeRestInfo*`) virtual
bool	_collide_shape(shape_rid: RID, transform: Transform3D, motion: Vector3, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool,

	results: `void*`, max_results: int, result_count: `int32_t*`) virtual
Vector3	_get_closest_point_to_object_volume(object: RID, point: Vector3) virtual const
int	_intersect_point(position: Vector3, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, results: `PhysicsServer3DExtensionShapeResult*`, max_results: int) virtual
bool	_intersect_ray(from: Vector3, to: Vector3, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, hit_from_inside: bool, hit_back_faces: bool, pick_ray: bool, result: `PhysicsServer3DExtensionRayResult*`) virtual
int	_intersect_shape(shape_rid: RID, transform: Transform3D, motion: Vector3, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, result_count: `PhysicsServer3DExtensionShapeResult*`, max_results: int) virtual
bool	_rest_info(shape_rid: RID, transform: Transform3D, motion: Vector3, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, rest_info: `PhysicsServer3DExtensionShapeRestInfo*`) virtual
bool	is_body_excluded_from_query(body: RID) const

Method Descriptions

bool **_cast_motion**(shape_rid: RID, transform: Transform3D, motion: Vector3, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, closest_safe: `float*`, closest_unsafe: `float*`, info: `PhysicsServer3DExtensionShapeRestInfo*`) virtual

There is currently no description for this method. Please help us by contributing one!

---

bool **_collide_shape**(shape_rid: RID, transform: Transform3D, motion: Vector3, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, results: `void*`, max_results: int, result_count: `int32_t*`) virtual

There is currently no description for this method. Please help us by contributing one!

Vector3 **_get_closest_point_to_object_volume**(object: RID, point: Vector3) virtual const

There is currently no description for this method. Please help us by contributing one!

---

int **_intersect_point**(position: Vector3, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, results: `PhysicsServer3DExtensionShapeResult*`, max_results: int) virtual

There is currently no description for this method. Please help us by contributing one!

---

bool **_intersect_ray**(from: Vector3, to: Vector3, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, hit_from_inside: bool, hit_back_faces: bool, pick_ray: bool, result: `PhysicsServer3DExtensionRayResult*`) virtual

There is currently no description for this method. Please help us by contributing one!

---

int **_intersect_shape**(shape_rid: RID, transform: Transform3D, motion: Vector3, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, result_count: `PhysicsServer3DExtensionShapeResult*`, max_results: int) virtual

There is currently no description for this method. Please help us by contributing one!

---

bool **_rest_info**(shape_rid: RID, transform: Transform3D, motion: Vector3, margin: float, collision_mask: int, collide_with_bodies: bool, collide_with_areas: bool, rest_info: `PhysicsServer3DExtensionShapeRestInfo*`) virtual

There is currently no description for this method. Please help us by contributing one!

---

bool **is_body_excluded_from_query**(body: RID) const

There is currently no description for this method. Please help us by contributing one!

PhysicsPointQueryParameters2D

Inherits: RefCounted **<** Object

Provides parameters for PhysicsDirectSpaceState2D.intersect_point.

Description

By changing various properties of this object, such as the point position, you can configure the parameters for PhysicsDirectSpaceState2D.intersect_point.

Properties

int	canvas_instance_id	`0`
bool	collide_with_areas	`false`
bool	collide_with_bodies	`true`
int	collision_mask	`4294967295`
Array[RID]	exclude	`[]`

Vector2	position	`Vector2(0, 0)`

Property Descriptions

int **canvas_instance_id** = `0` 🔗

- void **set_canvas_instance_id**(value: int)
- int **get_canvas_instance_id**()

If different from `0`, restricts the query to a specific canvas layer specified by its instance ID. See Object.get_instance_id.

If `0`, restricts the query to the Viewport's default canvas layer.

---

bool **collide_with_areas** = `false` 🔗

- void **set_collide_with_areas**(value: bool)
- bool **is_collide_with_areas_enabled**()

If `true`, the query will take Area2Ds into account.

---

bool **collide_with_bodies** = `true` 🔗

- void **set_collide_with_bodies**(value: bool)
- bool **is_collide_with_bodies_enabled**()

If `true`, the query will take PhysicsBody2Ds into account.

---

int **collision_mask** = `4294967295` 🔗

- void **set_collision_mask**(value: int)
- int **get_collision_mask**()

The physics layers the query will detect (as a bitmask). By default, all collision layers are detected. See Collision layers and masks in the documentation for more information.

Array[RID] **exclude** = `[]`

- void **set_exclude**(value: Array[RID])
- Array[RID] **get_exclude**()

The list of object RIDs that will be excluded from collisions. Use CollisionObject2D.get_rid to get the RID associated with a CollisionObject2D-derived node.

Note: The returned array is copied and any changes to it will not update the original property value. To update the value you need to modify the returned array, and then assign it to the property again.

Vector2 **position** = `Vector2(0, 0)`

- void **set_position**(value: Vector2)
- Vector2 **get_position**()

The position being queried for, in global coordinates.

PhysicsPointQueryParameters3D

Inherits: RefCounted **<** Object

Provides parameters for PhysicsDirectSpaceState3D.intersect_point.

Description

By changing various properties of this object, such as the point position, you can configure the parameters for PhysicsDirectSpaceState3D.intersect_point.

Properties

bool	collide_with_areas	`false`
bool	collide_with_bodies	`true`
int	collision_mask	`4294967295`
Array[RID]	exclude	`[]`
Vector3	position	`Vector3(0, 0, 0)`

Property Descriptions

bool **collide_with_areas** = `false` 🔗

- void **set_collide_with_areas**(value: bool)
- bool **is_collide_with_areas_enabled**()

If `true`, the query will take Area3Ds into account.

---

bool **collide_with_bodies** = `true` 🔗

- void **set_collide_with_bodies**(value: bool)
- bool **is_collide_with_bodies_enabled**()

If `true`, the query will take PhysicsBody3Ds into account.

---

int **collision_mask** = `4294967295` 🔗

- void **set_collision_mask**(value: int)
- int **get_collision_mask**()

The physics layers the query will detect (as a bitmask). By default, all collision layers are detected. See Collision layers and masks in the documentation for more information.

Array[RID] **exclude** = `[]` 🔗

- void **set_exclude**(value: Array[RID])
- Array[RID] **get_exclude**()

The list of object RIDs that will be excluded from collisions. Use CollisionObject3D.get_rid to get the RID associated with a CollisionObject3D-derived node.

Note: The returned array is copied and any changes to it will not update the original property value. To update the value you need to modify the returned array, and then assign it to the property again.

Vector3 **position** = `Vector3(0, 0, 0)` 🔗

- void **set_position**(value: Vector3)
- Vector3 **get_position**()

The position being queried for, in global coordinates.

PhysicsRayQueryParameters2D

Inherits: RefCounted **<** Object

Provides parameters for PhysicsDirectSpaceState2D.intersect_ray.

Description

By changing various properties of this object, such as the ray position, you can configure the parameters for PhysicsDirectSpaceState2D.intersect_ray.

Properties

bool	collide_with_areas	`false`
bool	collide_with_bodies	`true`
int	collision_mask	`4294967295`
Array[RID]	exclude	`[]`
Vector2	from	`Vector2(0, 0)`

bool	hit_from_inside	`false`
Vector2	to	`Vector2(0, 0)`

Methods

PhysicsRayQueryParameters2D	create(from: Vector2, to: Vector2, collision_mask: int = 4294967295, exclude: Array[RID] = []) static

---

Property Descriptions

bool **collide_with_areas** = `false`

- void **set_collide_with_areas**(value: bool)
- bool **is_collide_with_areas_enabled**()

If `true`, the query will take Area2Ds into account.

---

bool **collide_with_bodies** = `true`

- void **set_collide_with_bodies**(value: bool)
- bool **is_collide_with_bodies_enabled**()

If `true`, the query will take PhysicsBody2Ds into account.

---

int **collision_mask** = `4294967295`

- void **set_collision_mask**(value: int)
- int **get_collision_mask**()

The physics layers the query will detect (as a bitmask). By default, all collision layers are detected. See Collision layers and masks in the documentation for more information.

Array[RID] **exclude** = `[]` 🔗

- void **set_exclude**(value: Array[RID])
- Array[RID] **get_exclude**()

The list of object RIDs that will be excluded from collisions. Use CollisionObject2D.get_rid to get the RID associated with a CollisionObject2D-derived node.

Note: The returned array is copied and any changes to it will not update the original property value. To update the value you need to modify the returned array, and then assign it to the property again.

Vector2 **from** = `Vector2(0, 0)` 🔗

- void **set_from**(value: Vector2)
- Vector2 **get_from**()

The starting point of the ray being queried for, in global coordinates.

---

bool **hit_from_inside** = `false` 🔗

- void **set_hit_from_inside**(value: bool)
- bool **is_hit_from_inside_enabled**()

If `true`, the query will detect a hit when starting inside shapes. In this case the collision normal will be `Vector2(0, 0)`. Does not affect concave polygon shapes.

Vector2 **to** = `Vector2(0, 0)`

- void **set_to**(value: Vector2)
- Vector2 **get_to**()

The ending point of the ray being queried for, in global coordinates.

Method Descriptions

PhysicsRayQueryParameters2D **create**(from: Vector2, to: Vector2, collision_mask: int = 4294967295, exclude: Array[RID] = []) static

Returns a new, pre-configured **PhysicsRayQueryParameters2D** object. Use it to quickly create query parameters using the most common options.

```
var query = PhysicsRayQueryParameters2D.create(global_position, global_position + Vector2(0, 100))
var collision = get_world_2d().direct_space_state.intersect_ray(query)
```

PhysicsRayQueryParameters3D

Inherits: RefCounted **<** Object

Provides parameters for PhysicsDirectSpaceState3D.intersect_ray.

Description

By changing various properties of this object, such as the ray position, you can configure the parameters for PhysicsDirectSpaceState3D.intersect_ray.

Properties

bool	collide_with_areas	`false`
bool	collide_with_bodies	`true`
int	collision_mask	`4294967295`
Array[RID]	exclude	`[]`
Vector3	from	`Vector3(0, 0, 0)`

bool	hit_back_faces	`true`
bool	hit_from_inside	`false`
Vector3	to	`Vector3(0, 0, 0)`

Methods

PhysicsRayQueryParameters3D	create(from: Vector3, to: Vector3, collision_mask: int = 4294967295, exclude: Array[RID] = []) static

---

Property Descriptions

bool **collide_with_areas** = `false`

- void **set_collide_with_areas**(value: bool)
- bool **is_collide_with_areas_enabled**()

If `true`, the query will take Area3Ds into account.

---

bool **collide_with_bodies** = `true`

- void **set_collide_with_bodies**(value: bool)
- bool **is_collide_with_bodies_enabled**()

If `true`, the query will take PhysicsBody3Ds into account.

---

int **collision_mask** = `4294967295`

- void **set_collision_mask**(value: int)
- int **get_collision_mask**()

The physics layers the query will detect (as a bitmask). By default, all collision layers are detected. See Collision layers and masks in the documentation for more information.

Array[RID] **exclude** = `[]`

- void **set_exclude**(value: Array[RID])
- Array[RID] **get_exclude**()

The list of object RIDs that will be excluded from collisions. Use CollisionObject3D.get_rid to get the RID associated with a CollisionObject3D-derived node.

Note: The returned array is copied and any changes to it will not update the original property value. To update the value you need to modify the returned array, and then assign it to the property again.

Vector3 **from** = `Vector3(0, 0, 0)`

- void **set_from**(value: Vector3)
- Vector3 **get_from**()

The starting point of the ray being queried for, in global coordinates.

---

bool **hit_back_faces** = `true`

- void **set_hit_back_faces**(value: bool)
- bool **is_hit_back_faces_enabled**()

If `true`, the query will hit back faces with concave polygon shapes with back face enabled or heightmap shapes.

---

bool **hit_from_inside** = `false` 🔗

- void **set_hit_from_inside**(value: bool)
- bool **is_hit_from_inside_enabled**()

If `true`, the query will detect a hit when starting inside shapes. In this case the collision normal will be `Vector3(0, 0, 0)`. Does not affect concave polygon shapes or heightmap shapes.

Vector3 **to** = `Vector3(0, 0, 0)` 🔗

- void **set_to**(value: Vector3)
- Vector3 **get_to**()

The ending point of the ray being queried for, in global coordinates.

Method Descriptions

PhysicsRayQueryParameters3D **create**(from: Vector3, to: Vector3, collision_mask: int = 4294967295, exclude: Array[RID] = []) static 🔗

Returns a new, pre-configured **PhysicsRayQueryParameters3D** object. Use it to quickly create query parameters using the most common options.

```
var query = PhysicsRayQueryParameters3D.create(position,
position + Vector3(0, -10, 0))
var collision =
get_world_3d().direct_space_state.intersect_ray(query)
```

PhysicsServer2D

Inherits: Object

Inherited By: PhysicsServer2DExtension

A server interface for low-level 2D physics access.

Description

PhysicsServer2D is the server responsible for all 2D physics. It can directly create and manipulate all physics objects:

- A *space* is a self-contained world for a physics simulation. It contains bodies, areas, and joints. Its state can be queried for collision and intersection information, and several parameters of the simulation can be modified.
- A *shape* is a geometric shape such as a circle, a rectangle, a capsule, or a polygon. It can be used for collision detection by adding it to a body/area, possibly with an extra transformation relative to the body/area's origin. Bodies/areas can have multiple (transformed) shapes added to them, and a single shape can be added to bodies/areas multiple times with different local transformations.
- A *body* is a physical object which can be in static, kinematic, or rigid mode. Its state (such as position and velocity) can be queried and updated. A force integration callback can be set to customize the body's physics.
- An *area* is a region in space which can be used to detect bodies and areas entering and exiting it. A body monitoring callback can be set to report entering/exiting body shapes, and similarly an area monitoring callback can be set. Gravity and damping can be overridden within the area by setting area parameters.
- A *joint* is a constraint, either between two bodies or on one body relative to a point. Parameters such as the joint bias and the rest length of a spring joint can be adjusted.

Physics objects in **PhysicsServer2D** may be created and manipulated independently; they do not have to be tied to nodes in the scene tree.

Note: All the 2D physics nodes use the physics server internally. Adding a physics node to the scene tree will cause a corresponding physics object to be created in the physics server. A rigid body node registers a callback that updates the node's transform with the transform of the respective body object in the physics server (every physics update). An area node registers a callback to inform the area node about overlaps with the respective area object in the physics server. The raycast node queries the direct state of the relevant space in the physics server.

Methods

void	area_add_shape(area: RID, shape: RID, transform: Transform2D = Transform2D(1, 0, 0, 1, 0, 0), disabled: bool = false)
void	area_attach_canvas_instance_id(area: RID, id: int)
void	area_attach_object_instance_id(area: RID, id: int)
void	area_clear_shapes(area: RID)
RID	area_create()
int	area_get_canvas_instance_id(area: RID) const
int	area_get_collision_layer(area: RID) const
int	area_get_collision_mask(area: RID) const
int	area_get_object_instance_id(area: RID) const
Variant	area_get_param(area: RID, param: AreaParameter) const
RID	area_get_shape(area: RID, shape_idx: int) const

int	area_get_shape_count(area: RID) const
Transform2D	area_get_shape_transform(area: RID, shape_idx: int) const
RID	area_get_space(area: RID) const
Transform2D	area_get_transform(area: RID) const
void	area_remove_shape(area: RID, shape_idx: int)
void	area_set_area_monitor_callback(area: RID, callback: Callable)
void	area_set_collision_layer(area: RID, layer: int)
void	area_set_collision_mask(area: RID, mask: int)
void	area_set_monitor_callback(area: RID, callback: Callable)
void	area_set_monitorable(area: RID, monitorable: bool)
void	area_set_param(area: RID, param: AreaParameter, value: Variant)
void	area_set_shape(area: RID, shape_idx: int, shape: RID)
void	area_set_shape_disabled(area: RID, shape_idx: int, disabled: bool)

void	area_set_shape_transform(area: RID, shape_idx: int, transform: Transform2D)
void	area_set_space(area: RID, space: RID)
void	area_set_transform(area: RID, transform: Transform2D)
void	body_add_collision_exception(body: RID, excepted_body: RID)
void	body_add_constant_central_force(body: RID, force: Vector2)
void	body_add_constant_force(body: RID, force: Vector2, position: Vector2 = Vector2(0, 0))
void	body_add_constant_torque(body: RID, torque: float)
void	body_add_shape(body: RID, shape: RID, transform: Transform2D = Transform2D(1, 0, 0, 1, 0, 0), disabled: bool = false)
void	body_apply_central_force(body: RID, force: Vector2)
void	body_apply_central_impulse(body: RID, impulse: Vector2)
void	body_apply_force(body: RID, force: Vector2, position: Vector2 = Vector2(0, 0))
void	body_apply_impulse(body: RID, impulse: Vector2, position: Vector2 = Vector2(0, 0))

void	body_apply_torque(body: RID, torque: float)
void	body_apply_torque_impulse(body: RID, impulse: float)
void	body_attach_canvas_instance_id(body: RID, id: int)
void	body_attach_object_instance_id(body: RID, id: int)
void	body_clear_shapes(body: RID)
RID	body_create()
int	body_get_canvas_instance_id(body: RID) const
int	body_get_collision_layer(body: RID) const
int	body_get_collision_mask(body: RID) const
float	body_get_collision_priority(body: RID) const
Vector2	body_get_constant_force(body: RID) const
float	body_get_constant_torque(body: RID) const
CCDMode	body_get_continuous_collision_detection_mode(body: RID) const
PhysicsDirectBodyState2D	body_get_direct_state(body: RID)
int	body_get_max_contacts_reported(body: RID) const

BodyMode	body_get_mode(body: RID) const
int	body_get_object_instance_id(body: RID) const
Variant	body_get_param(body: RID, param: BodyParameter) const
RID	body_get_shape(body: RID, shape_idx: int) const
int	body_get_shape_count(body: RID) const
Transform2D	body_get_shape_transform(body: RID, shape_idx: int) const
RID	body_get_space(body: RID) const
Variant	body_get_state(body: RID, state: BodyState) const
bool	body_is_omitting_force_integration(body: RID) const
void	body_remove_collision_exception(body: RID, excepted_body: RID)
void	body_remove_shape(body: RID, shape_idx: int)
void	body_reset_mass_properties(body: RID)
void	body_set_axis_velocity(body: RID, axis_velocity: Vector2)
void	body_set_collision_layer(body: RID, layer: int)

void	body_set_collision_mask(body: RID, mask: int)
void	body_set_collision_priority(body: RID, priority: float)
void	body_set_constant_force(body: RID, force: Vector2)
void	body_set_constant_torque(body: RID, torque: float)
void	body_set_continuous_collision_detection_mode(body: RID, mode: CCDMode)
void	body_set_force_integration_callback(body: RID, callable: Callable, userdata: Variant = null)
void	body_set_max_contacts_reported(body: RID, amount: int)
void	body_set_mode(body: RID, mode: BodyMode)
void	body_set_omit_force_integration(body: RID, enable: bool)
void	body_set_param(body: RID, param: BodyParameter, value: Variant)
void	body_set_shape(body: RID, shape_idx: int, shape: RID)
void	body_set_shape_as_one_way_collision(body: RID, shape_idx: int, enable: bool, margin: float)

void	body_set_shape_disabled(body: RID, shape_idx: int, disabled: bool)
void	body_set_shape_transform(body: RID, shape_idx: int, transform: Transform2D)
void	body_set_space(body: RID, space: RID)
void	body_set_state(body: RID, state: BodyState, value: Variant)
void	body_set_state_sync_callback(body: RID, callable: Callable)
bool	body_test_motion(body: RID, parameters: PhysicsTestMotionParameters2D, result: PhysicsTestMotionResult2D = null)
RID	capsule_shape_create()
RID	circle_shape_create()
RID	concave_polygon_shape_create()
RID	convex_polygon_shape_create()
float	damped_spring_joint_get_param(joint: RID, param: DampedSpringParam) const
void	damped_spring_joint_set_param(joint: RID, param: DampedSpringParam, value: float)

void	free_rid(rid: RID)
int	get_process_info(process_info: ProcessInfo)
void	joint_clear(joint: RID)
RID	joint_create()
void	joint_disable_collisions_between_bodies(joint: RID, disable: bool)
float	joint_get_param(joint: RID, param: JointParam) const
JointType	joint_get_type(joint: RID) const
bool	joint_is_disabled_collisions_between_bodies(joint: RID) const
void	joint_make_damped_spring(joint: RID, anchor_a: Vector2, anchor_b: Vector2, body_a: RID, body_b: RID = RID())
void	joint_make_groove(joint: RID, groove1_a: Vector2, groove2_a: Vector2, anchor_b: Vector2, body_a: RID = RID(), body_b: RID = RID())
void	joint_make_pin(joint: RID, anchor: Vector2, body_a: RID, body_b: RID = RID())
void	joint_set_param(joint: RID, param: JointParam, value: float)

bool	pin_joint_get_flag(joint: RID, flag: PinJointFlag) const
float	pin_joint_get_param(joint: RID, param: PinJointParam) const
void	pin_joint_set_flag(joint: RID, flag: PinJointFlag, enabled: bool)
void	pin_joint_set_param(joint: RID, param: PinJointParam, value: float)
RID	rectangle_shape_create()
RID	segment_shape_create()
RID	separation_ray_shape_create()
void	set_active(active: bool)
Variant	shape_get_data(shape: RID) const
ShapeType	shape_get_type(shape: RID) const
void	shape_set_data(shape: RID, data: Variant)
RID	space_create()
PhysicsDirectSpaceState2D	space_get_direct_state(space: RID)
float	space_get_param(space: RID, param: SpaceParameter) const

bool	space_is_active(space: RID) const
void	space_set_active(space: RID, active: bool)
void	space_set_param(space: RID, param: SpaceParameter, value: float)
RID	world_boundary_shape_create()

Enumerations

enum **SpaceParameter**: 🔗

SpaceParameter **SPACE_PARAM_CONTACT_RECYCLE_RADIUS** = `0`

Constant to set/get the maximum distance a pair of bodies has to move before their collision status has to be recalculated. The default value of this parameter is ProjectSettings.physics/2d/solver/contact_recycle_radius.

SpaceParameter **SPACE_PARAM_CONTACT_MAX_SEPARATION** = `1`

Constant to set/get the maximum distance a shape can be from another before they are considered separated and the contact is discarded. The default value of this parameter is ProjectSettings.physics/2d/solver/contact_max_separation.

SpaceParameter **SPACE_PARAM_CONTACT_MAX_ALLOWED_PENETRATION** = `2`

Constant to set/get the maximum distance a shape can penetrate another shape before it is considered a collision. The default value of this parameter is ProjectSettings.physics/2d/solver/contact_max_allowed_penetration.

SpaceParameter **SPACE_PARAM_CONTACT_DEFAULT_BIAS** = `3`

Constant to set/get the default solver bias for all physics contacts. A solver bias is a factor controlling how much two objects "rebound", after overlapping, to avoid leaving them in that state because of numerical imprecision. The default value of this parameter is ProjectSettings.physics/2d/solver/default_contact_bias.

SpaceParameter **SPACE_PARAM_BODY_LINEAR_VELOCITY_SLEEP_THRESHOLD** = `4`

Constant to set/get the threshold linear velocity of activity. A body marked as potentially inactive for both linear and angular velocity will be put to sleep after the time given. The default value of this parameter is ProjectSettings.physics/2d/sleep_threshold_linear.

SpaceParameter **SPACE_PARAM_BODY_ANGULAR_VELOCITY_SLEEP_THRESHOLD** = `5`

Constant to set/get the threshold angular velocity of activity. A body marked as potentially inactive for both linear and angular velocity will be put to sleep after the time given. The default value of this parameter is ProjectSettings.physics/2d/sleep_threshold_angular.

SpaceParameter **SPACE_PARAM_BODY_TIME_TO_SLEEP** = `6`

Constant to set/get the maximum time of activity. A body marked as potentially inactive for both linear and angular velocity will be put to sleep after this time. The default value of this parameter is ProjectSettings.physics/2d/time_before_sleep.

SpaceParameter **SPACE_PARAM_CONSTRAINT_DEFAULT_BIAS** = `7`

Constant to set/get the default solver bias for all physics constraints. A solver bias is a factor controlling how much two objects "rebound", after violating a constraint, to avoid leaving them in that state because of numerical imprecision. The default value of this parameter is ProjectSettings.physics/2d/solver/default_constraint_bias.

SpaceParameter **SPACE_PARAM_SOLVER_ITERATIONS** = `8`

Constant to set/get the number of solver iterations for all contacts and constraints. The greater the number of iterations, the more accurate the collisions will be. However, a greater number of iterations requires more CPU power, which can decrease performance. The default value of this parameter is ProjectSettings.physics/2d/solver/solver_iterations.

---

enum **ShapeType**: 🔗

ShapeType **SHAPE_WORLD_BOUNDARY** = `0`

This is the constant for creating world boundary shapes. A world boundary shape is an *infinite* line with an origin point, and a normal. Thus, it can be used for front/behind checks.

ShapeType **SHAPE_SEPARATION_RAY** = 1

This is the constant for creating separation ray shapes. A separation ray is defined by a length and separates itself from what is touching its far endpoint. Useful for character controllers.

ShapeType **SHAPE_SEGMENT** = 2

This is the constant for creating segment shapes. A segment shape is a *finite* line from a point A to a point B. It can be checked for intersections.

ShapeType **SHAPE_CIRCLE** = 3

This is the constant for creating circle shapes. A circle shape only has a radius. It can be used for intersections and inside/outside checks.

ShapeType **SHAPE_RECTANGLE** = 4

This is the constant for creating rectangle shapes. A rectangle shape is defined by a width and a height. It can be used for intersections and inside/outside checks.

ShapeType **SHAPE_CAPSULE** = 5

This is the constant for creating capsule shapes. A capsule shape is defined by a radius and a length. It can be used for intersections and inside/outside checks.

ShapeType **SHAPE_CONVEX_POLYGON** = 6

This is the constant for creating convex polygon shapes. A polygon is defined by a list of points. It can be used for intersections and inside/outside checks.

ShapeType **SHAPE_CONCAVE_POLYGON** = 7

This is the constant for creating concave polygon shapes. A polygon is defined by a list of points. It can be used for intersections checks, but not for inside/outside checks.

ShapeType **SHAPE_CUSTOM** = 8

This constant is used internally by the engine. Any attempt to create this kind of shape results in an error.

---

enum **AreaParameter**: 🔗

AreaParameter **AREA_PARAM_GRAVITY_OVERRIDE_MODE** = `0`

Constant to set/get gravity override mode in an area. See AreaSpaceOverrideMode for possible values. The default value of this parameter is AREA_SPACE_OVERRIDE_DISABLED.

AreaParameter **AREA_PARAM_GRAVITY** = `1`

Constant to set/get gravity strength in an area. The default value of this parameter is `9.80665`.

AreaParameter **AREA_PARAM_GRAVITY_VECTOR** = `2`

Constant to set/get gravity vector/center in an area. The default value of this parameter is `Vector2(0, -1)`.

AreaParameter **AREA_PARAM_GRAVITY_IS_POINT** = `3`

Constant to set/get whether the gravity vector of an area is a direction, or a center point. The default value of this parameter is `false`.

AreaParameter **AREA_PARAM_GRAVITY_POINT_UNIT_DISTANCE** = `4`

Constant to set/get the distance at which the gravity strength is equal to the gravity controlled by AREA_PARAM_GRAVITY. For example, on a planet 100 pixels in radius with a surface gravity of 4.0 px/s², set the gravity to 4.0 and the unit distance to 100.0. The gravity will have falloff according to the inverse square law, so in the example, at 200 pixels from the center the gravity will be 1.0 px/s² (twice the distance, 1/4th the gravity), at 50 pixels it will be 16.0 px/s² (half the distance, 4x the gravity), and so on.

The above is true only when the unit distance is a positive number. When the unit distance is set to 0.0, the gravity will be constant regardless of distance. The default value of this parameter is `0.0`.

AreaParameter **AREA_PARAM_LINEAR_DAMP_OVERRIDE_MODE** = `5`

Constant to set/get linear damping override mode in an area. See AreaSpaceOverrideMode for possible values. The default value of this parameter is AREA_SPACE_OVERRIDE_DISABLED.

AreaParameter **AREA_PARAM_LINEAR_DAMP** = `6`

Constant to set/get the linear damping factor of an area. The default value of this parameter is `0.1`.

AreaParameter **AREA_PARAM_ANGULAR_DAMP_OVERRIDE_MODE** = `7`

Constant to set/get angular damping override mode in an area. See AreaSpaceOverrideMode for possible values. The default value of this parameter is AREA_SPACE_OVERRIDE_DISABLED.

AreaParameter **AREA_PARAM_ANGULAR_DAMP** = `8`

Constant to set/get the angular damping factor of an area. The default value of this parameter is `1.0`.

AreaParameter **AREA_PARAM_PRIORITY** = `9`

Constant to set/get the priority (order of processing) of an area. The default value of this parameter is `0`.

---

enum **AreaSpaceOverrideMode**: 🔗

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_DISABLED** = `0`

This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_COMBINE** = `1`

This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_COMBINE_REPLACE** = `2`

This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_REPLACE** = `3`

This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_REPLACE_COMBINE** = `4`

This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.

---

enum **BodyMode**: 🔗

BodyMode **BODY_MODE_STATIC** = `0`

Constant for static bodies. In this mode, a body can be only moved by user code and doesn't collide with other bodies along its path when moved.

BodyMode **BODY_MODE_KINEMATIC** = `1`

Constant for kinematic bodies. In this mode, a body can be only moved by user code and collides with other bodies along its path.

BodyMode **BODY_MODE_RIGID** = `2`

Constant for rigid bodies. In this mode, a body can be pushed by other bodies and has forces applied.

BodyMode **BODY_MODE_RIGID_LINEAR** = `3`

Constant for linear rigid bodies. In this mode, a body can not rotate, and only its linear velocity is affected by external forces.

---

enum **BodyParameter**: 🔗

BodyParameter **BODY_PARAM_BOUNCE** = `0`

Constant to set/get a body's bounce factor. The default value of this parameter is `0.0`.

BodyParameter **BODY_PARAM_FRICTION** = `1`

Constant to set/get a body's friction. The default value of this parameter is `1.0`.

BodyParameter **BODY_PARAM_MASS** = `2`

Constant to set/get a body's mass. The default value of this parameter is `1.0`. If the body's mode is set to BODY_MODE_RIGID, then setting this parameter will have the following additional effects:

- If the parameter BODY_PARAM_CENTER_OF_MASS has never been set explicitly, then the value of that parameter will be recalculated based on the body's shapes.
- If the parameter BODY_PARAM_INERTIA is set to a value `<= 0.0`, then the value of that parameter will be recalculated based on the body's shapes, mass, and center of mass.

BodyParameter **BODY_PARAM_INERTIA** = `3`

Constant to set/get a body's inertia. The default value of this parameter is `0.0`. If the body's inertia is set to a value `<= 0.0`, then the inertia will be recalculated based on the body's shapes, mass, and center of mass.

BodyParameter **BODY_PARAM_CENTER_OF_MASS** = `4`

Constant to set/get a body's center of mass position in the body's local coordinate system. The default value of this parameter is `Vector2(0,0)`. If this parameter is never set explicitly, then it is recalculated based on the body's shapes when setting the parameter BODY_PARAM_MASS or when calling body_set_space.

BodyParameter **BODY_PARAM_GRAVITY_SCALE** = `5`

Constant to set/get a body's gravity multiplier. The default value of this parameter is `1.0`.

BodyParameter **BODY_PARAM_LINEAR_DAMP_MODE** = `6`

Constant to set/get a body's linear damping mode. See BodyDampMode for possible values. The default value of this parameter is BODY_DAMP_MODE_COMBINE.

BodyParameter **BODY_PARAM_ANGULAR_DAMP_MODE** = `7`

Constant to set/get a body's angular damping mode. See BodyDampMode for possible values. The default value of this parameter is BODY_DAMP_MODE_COMBINE.

BodyParameter **BODY_PARAM_LINEAR_DAMP** = `8`

Constant to set/get a body's linear damping factor. The default value of this parameter is `0.0`.

BodyParameter **BODY_PARAM_ANGULAR_DAMP** = `9`

Constant to set/get a body's angular damping factor. The default value of this parameter is `0.0`.

BodyParameter **BODY_PARAM_MAX** = `10`

Represents the size of the BodyParameter enum.

---

enum **BodyDampMode**: 🔗

BodyDampMode **BODY_DAMP_MODE_COMBINE** = `0`

The body's damping value is added to any value set in areas or the default value.

BodyDampMode **BODY_DAMP_MODE_REPLACE** = `1`

The body's damping value replaces any value set in areas or the default value.

---

enum **BodyState**: 🔗

BodyState **BODY_STATE_TRANSFORM** = `0`

Constant to set/get the current transform matrix of the body.

BodyState **BODY_STATE_LINEAR_VELOCITY** = `1`

Constant to set/get the current linear velocity of the body.

BodyState **BODY_STATE_ANGULAR_VELOCITY** = `2`

Constant to set/get the current angular velocity of the body.

BodyState **BODY_STATE_SLEEPING** = `3`

Constant to sleep/wake up a body, or to get whether it is sleeping.

BodyState **BODY_STATE_CAN_SLEEP** = `4`

Constant to set/get whether the body can sleep.

---

enum **JointType**: 🔗

JointType **JOINT_TYPE_PIN** = `0`

Constant to create pin joints.

JointType **JOINT_TYPE_GROOVE** = `1`

Constant to create groove joints.

JointType **JOINT_TYPE_DAMPED_SPRING** = `2`

Constant to create damped spring joints.

JointType **JOINT_TYPE_MAX** = `3`

Represents the size of the JointType enum.

enum **JointParam**: 🔗

JointParam **JOINT_PARAM_BIAS** = `0`

Constant to set/get how fast the joint pulls the bodies back to satisfy the joint constraint. The lower the value, the more the two bodies can pull on the joint. The default value of this parameter is `0.0`.

Note: In Godot Physics, this parameter is only used for pin joints and groove joints.

JointParam **JOINT_PARAM_MAX_BIAS** = `1`

Constant to set/get the maximum speed with which the joint can apply corrections. The default value of this parameter is `3.40282e+38`.

Note: In Godot Physics, this parameter is only used for groove joints.

JointParam **JOINT_PARAM_MAX_FORCE** = `2`

Constant to set/get the maximum force that the joint can use to act on the two bodies. The default value of this parameter is `3.40282e+38`.

Note: In Godot Physics, this parameter is only used for groove joints.

enum **PinJointParam**: 🔗

PinJointParam **PIN_JOINT_SOFTNESS** = `0`

Constant to set/get a how much the bond of the pin joint can flex. The default value of this parameter is `0.0`.

PinJointParam **PIN_JOINT_LIMIT_UPPER** = `1`

The maximum rotation around the pin.

PinJointParam **PIN_JOINT_LIMIT_LOWER** = `2`

The minimum rotation around the pin.

PinJointParam **PIN_JOINT_MOTOR_TARGET_VELOCITY** = `3`

Target speed for the motor. In radians per second.

---

enum **PinJointFlag**: 🔗

PinJointFlag **PIN_JOINT_FLAG_ANGULAR_LIMIT_ENABLED** = `0`

If `true`, the pin has a maximum and a minimum rotation.

PinJointFlag **PIN_JOINT_FLAG_MOTOR_ENABLED** = `1`

If `true`, a motor turns the pin.

---

enum **DampedSpringParam**: 🔗

DampedSpringParam **DAMPED_SPRING_REST_LENGTH** = `0`

Sets the resting length of the spring joint. The joint will always try to go to back this length when pulled apart. The default value of this parameter is the distance between the joint's anchor points.

DampedSpringParam **DAMPED_SPRING_STIFFNESS** = `1`

Sets the stiffness of the spring joint. The joint applies a force equal to the stiffness times the distance from its resting length. The default value of this parameter is `20.0`.

DampedSpringParam **DAMPED_SPRING_DAMPING** = `2`

Sets the damping ratio of the spring joint. A value of 0 indicates an undamped spring, while 1 causes the system to reach equilibrium as fast as possible (critical damping). The default value of this parameter is `1.5`.

---

enum **CCDMode**:

CCDMode **CCD_MODE_DISABLED** = `0`

Disables continuous collision detection. This is the fastest way to detect body collisions, but it can miss small and/or fast-moving objects.

CCDMode **CCD_MODE_CAST_RAY** = `1`

Enables continuous collision detection by raycasting. It is faster than shapecasting, but less precise.

CCDMode **CCD_MODE_CAST_SHAPE** = `2`

Enables continuous collision detection by shapecasting. It is the slowest CCD method, and the most precise.

---

enum **AreaBodyStatus**:

AreaBodyStatus **AREA_BODY_ADDED** = `0`

The value of the first parameter and area callback function receives, when an object enters one of its shapes.

AreaBodyStatus **AREA_BODY_REMOVED** = `1`

The value of the first parameter and area callback function receives, when an object exits one of its shapes.

---

enum **ProcessInfo**:

ProcessInfo **INFO_ACTIVE_OBJECTS** = `0`

Constant to get the number of objects that are not sleeping.

ProcessInfo **INFO_COLLISION_PAIRS** = `1`

Constant to get the number of possible collisions.

ProcessInfo **INFO_ISLAND_COUNT** = `2`

Constant to get the number of space regions where a collision could occur.

Method Descriptions

void **area_add_shape**(area: RID, shape: RID, transform: Transform2D = Transform2D(1, 0, 0, 1, 0, 0), disabled: bool = false)

Adds a shape to the area, with the given local transform. The shape (together with its `transform` and `disabled` properties) is added to an array of shapes, and the shapes of an area are usually referenced by their index in this array.

void **area_attach_canvas_instance_id**(area: RID, id: int)

Attaches the `ObjectID` of a canvas to the area. Use Object.get_instance_id to get the `ObjectID` of a CanvasLayer.

void **area_attach_object_instance_id**(area: RID, id: int)

Attaches the `ObjectID` of an Object to the area. Use Object.get_instance_id to get the `ObjectID` of a CollisionObject2D.

void **area_clear_shapes**(area: RID)

Removes all shapes from the area. This does not delete the shapes themselves, so they can continue to be used elsewhere or added back later.

RID **area_create**()

Creates a 2D area object in the physics server, and returns the RID that identifies it. The default settings for the created area include a collision layer and mask set to `1`, and `monitorable` set to `false`.

Use area_add_shape to add shapes to it, use area_set_transform to set its transform, and use area_set_space to add the area to a space. If you want the area to be detectable use area_set_monitorable.

int **area_get_canvas_instance_id**(area: RID) const

Returns the `ObjectID` of the canvas attached to the area. Use @GlobalScope.instance_from_id to retrieve a CanvasLayer from a nonzero `ObjectID`.

---

int **area_get_collision_layer**(area: RID) const

Returns the physics layer or layers the area belongs to, as a bitmask.

---

int **area_get_collision_mask**(area: RID) const

Returns the physics layer or layers the area can contact with, as a bitmask.

---

int **area_get_object_instance_id**(area: RID) const

Returns the `ObjectID` attached to the area. Use @GlobalScope.instance_from_id to retrieve an Object from a nonzero `ObjectID`.

Variant **area_get_param**(area: RID, param: AreaParameter) const

Returns the value of the given area parameter. See AreaParameter for the list of available parameters.

RID **area_get_shape**(area: RID, shape_idx: int) const

Returns the RID of the shape with the given index in the area's array of shapes.

---

int **area_get_shape_count**(area: RID) const

Returns the number of shapes added to the area.

Transform2D **area_get_shape_transform**(area: RID, shape_idx: int) const

Returns the local transform matrix of the shape with the given index in the area's array of shapes.

RID **area_get_space**(area: RID) const

Returns the RID of the space assigned to the area. Returns an empty RID if no space is assigned.

Transform2D **area_get_transform**(area: RID) const

Returns the transform matrix of the area.

---

void **area_remove_shape**(area: RID, shape_idx: int)

Removes the shape with the given index from the area's array of shapes. The shape itself is not deleted, so it can continue to be used elsewhere or added back later. As a result of this operation, the area's shapes which used to have indices higher than `shape_idx` will have their index decreased by one.

---

void **area_set_area_monitor_callback**(area: RID, callback: Callable)

Sets the area's area monitor callback. This callback will be called when any other (shape of an) area enters or exits (a shape of) the given area, and must take the following five parameters:

1. an integer `status`: either AREA_BODY_ADDED or AREA_BODY_REMOVED depending on whether the other area's shape entered or exited the area,
2. an RID `area_rid`: the RID of the other area that entered or exited the area,
3. an integer `instance_id`: the `ObjectID` attached to the other area,
4. an integer `area_shape_idx`: the index of the shape of the other area that entered or exited the area,
5. an integer `self_shape_idx`: the index of the shape of the area where the other area entered or exited.

By counting (or keeping track of) the shapes that enter and exit, it can be determined if an area (with all its shapes) is entering for the first time or exiting for the last time.

---

void **area_set_collision_layer**(area: RID, layer: int)

Assigns the area to one or many physics layers, via a bitmask.

---

void **area_set_collision_mask**(area: RID, mask: int)

Sets which physics layers the area will monitor, via a bitmask.

---

void **area_set_monitor_callback**(area: RID, callback: Callable)

Sets the area's body monitor callback. This callback will be called when any other (shape of a) body enters or exits (a shape of) the given area, and must take the following five parameters:

1. an integer `status`: either AREA_BODY_ADDED or AREA_BODY_REMOVED depending on whether the other body shape entered or exited the area,
2. an RID `body_rid`: the RID of the body that entered or exited the area,
3. an integer `instance_id`: the `ObjectID` attached to the body,
4. an integer `body_shape_idx`: the index of the shape of the body that entered or exited the area,
5. an integer `self_shape_idx`: the index of the shape of the area where the body entered or exited.

By counting (or keeping track of) the shapes that enter and exit, it can be determined if a body (with all its shapes) is entering for the first time or exiting for the last time.

---

void **area_set_monitorable**(area: RID, monitorable: bool)

Sets whether the area is monitorable or not. If `monitorable` is `true`, the area monitoring callback of other areas will be called when this area enters or exits them.

---

void **area_set_param**(area: RID, param: AreaParameter, value: Variant)

Sets the value of the given area parameter. See AreaParameter for the list of available parameters.

---

void **area_set_shape**(area: RID, shape_idx: int, shape: RID)

Replaces the area's shape at the given index by another shape, while not affecting the `transform` and `disabled` properties at the same index.

---

void **area_set_shape_disabled**(area: RID, shape_idx: int, disabled: bool)

Sets the disabled property of the area's shape with the given index. If `disabled` is `true`, then the shape will not detect any other shapes entering or exiting it.

---

void **area_set_shape_transform**(area: RID, shape_idx: int, transform: Transform2D) 🔗

Sets the local transform matrix of the area's shape with the given index.

---

void **area_set_space**(area: RID, space: RID) 🔗

Adds the area to the given space, after removing the area from the previously assigned space (if any).

Note: To remove an area from a space without immediately adding it back elsewhere, use `PhysicsServer2D.area_set_space(area, RID())`.

---

void **area_set_transform**(area: RID, transform: Transform2D) 🔗

Sets the transform matrix of the area.

---

void **body_add_collision_exception**(body: RID, excepted_body: RID) 🔗

Adds `excepted_body` to the body's list of collision exceptions, so that collisions with it are ignored.

---

void **body_add_constant_central_force**(body: RID, force: Vector2) 🔗

Adds a constant directional force to the body. The force does not affect rotation. The force remains applied over time until cleared with `PhysicsServer2D.body_set_constant_force(body, Vector2(0, 0))`.

This is equivalent to using body_add_constant_force at the body's center of mass.

---

void **body_add_constant_force**(body: RID, force: Vector2, position: Vector2 = Vector2(0, 0)) 🔗

Adds a constant positioned force to the body. The force can affect rotation if `position` is different from the body's center of mass. The force remains applied over time until cleared with `PhysicsServer2D.body_set_constant_force(body, Vector2(0, 0))`.

`position` is the offset from the body origin in global coordinates.

---

void **body_add_constant_torque**(body: RID, torque: float)

Adds a constant rotational force to the body. The force does not affect position. The force remains applied over time until cleared with `PhysicsServer2D.body_set_constant_torque(body, 0)`.

---

void **body_add_shape**(body: RID, shape: RID, transform: Transform2D = Transform2D(1, 0, 0, 1, 0, 0), disabled: bool = false)

Adds a shape to the area, with the given local transform. The shape (together with its `transform` and `disabled` properties) is added to an array of shapes, and the shapes of a body are usually referenced by their index in this array.

---

void **body_apply_central_force**(body: RID, force: Vector2)

Applies a directional force to the body, at the body's center of mass. The force does not affect rotation. A force is time dependent and meant to be applied every physics update.

This is equivalent to using body_apply_force at the body's center of mass.

---

void **body_apply_central_impulse**(body: RID, impulse: Vector2)

Applies a directional impulse to the body, at the body's center of mass. The impulse does not affect rotation.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

This is equivalent to using body_apply_impulse at the body's center of mass.

---

void **body_apply_force**(body: RID, force: Vector2, position: Vector2 = Vector2(0, 0))

Applies a positioned force to the body. The force can affect rotation if `position` is different from the body's center of mass. A force is time dependent and meant to be applied every physics update.

`position` is the offset from the body origin in global coordinates.

---

void **body_apply_impulse**(body: RID, impulse: Vector2, position: Vector2 = Vector2(0, 0))

Applies a positioned impulse to the body. The impulse can affect rotation if `position` is different from the body's center of mass.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

`position` is the offset from the body origin in global coordinates.

---

void **body_apply_torque**(body: RID, torque: float)

Applies a rotational force to the body. The force does not affect position. A force is time dependent and meant to be applied every physics update.

---

void **body_apply_torque_impulse**(body: RID, impulse: float)

Applies a rotational impulse to the body. The impulse does not affect position.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

---

void **body_attach_canvas_instance_id**(body: RID, id: int)

Attaches the `ObjectID` of a canvas to the body. Use Object.get_instance_id to get the `ObjectID` of a CanvasLayer.

---

void **body_attach_object_instance_id**(body: RID, id: int)

Attaches the `ObjectID` of an Object to the body. Use Object.get_instance_id to get the `ObjectID` of a CollisionObject2D.

void **body_clear_shapes**(body: RID)

Removes all shapes from the body. This does not delete the shapes themselves, so they can continue to be used elsewhere or added back later.

RID **body_create**()

Creates a 2D body object in the physics server, and returns the RID that identifies it. The default settings for the created area include a collision layer and mask set to `1`, and body mode set to BODY_MODE_RIGID.

Use body_add_shape to add shapes to it, use body_set_state to set its transform, and use body_set_space to add the body to a space.

int **body_get_canvas_instance_id**(body: RID) const

Returns the `ObjectID` of the canvas attached to the body. Use @GlobalScope.instance_from_id to retrieve a CanvasLayer from a nonzero `ObjectID`.

int **body_get_collision_layer**(body: RID) const

Returns the physics layer or layers the body belongs to, as a bitmask.

int **body_get_collision_mask**(body: RID) const

Returns the physics layer or layers the body can collide with, as a bitmask.

float **body_get_collision_priority**(body: RID) const

Returns the body's collision priority. This is used in the depenetration phase of body_test_motion. The higher the priority is, the lower the penetration into the body will be.

Vector2 **body_get_constant_force**(body: RID) const

Returns the body's total constant positional force applied during each physics update.

See body_add_constant_force and body_add_constant_central_force.

---

float **body_get_constant_torque**(body: RID) const

Returns the body's total constant rotational force applied during each physics update.

See body_add_constant_torque.

CCDMode **body_get_continuous_collision_detection_mode**(body: RID) const

Returns the body's continuous collision detection mode (see CCDMode).

PhysicsDirectBodyState2D **body_get_direct_state**(body: RID)

Returns the PhysicsDirectBodyState2D of the body. Returns `null` if the body is destroyed or not assigned to a space.

---

int **body_get_max_contacts_reported**(body: RID) const

Returns the maximum number of contacts that the body can report. See body_set_max_contacts_reported.

BodyMode **body_get_mode**(body: RID) const

Returns the body's mode (see BodyMode).

---

int **body_get_object_instance_id**(body: RID) const

Returns the `ObjectID` attached to the body. Use @GlobalScope.instance_from_id to retrieve an Object from a nonzero `ObjectID`.

Variant **body_get_param**(body: RID, param: BodyParameter) const

Returns the value of the given body parameter. See BodyParameter for the list of available parameters.

RID **body_get_shape**(body: RID, shape_idx: int) const

Returns the RID of the shape with the given index in the body's array of shapes.

---

int **body_get_shape_count**(body: RID) const

Returns the number of shapes added to the body.

Transform2D **body_get_shape_transform**(body: RID, shape_idx: int) const

Returns the local transform matrix of the shape with the given index in the area's array of shapes.

RID **body_get_space**(body: RID) const

Returns the RID of the space assigned to the body. Returns an empty RID if no space is assigned.

Variant **body_get_state**(body: RID, state: BodyState) const

Returns the value of the given state of the body. See BodyState for the list of available states.

---

bool **body_is_omitting_force_integration**(body: RID) const

Returns `true` if the body is omitting the standard force integration. See body_set_omit_force_integration.

---

void **body_remove_collision_exception**(body: RID, excepted_body: RID)

Removes `excepted_body` from the body's list of collision exceptions, so that collisions with it are no longer ignored.

void **body_remove_shape**(body: RID, shape_idx: int)

Removes the shape with the given index from the body's array of shapes. The shape itself is not deleted, so it can continue to be used elsewhere or added back later. As a result of this operation, the body's shapes which used to have indices higher than `shape_idx` will have their index decreased by one.

void **body_reset_mass_properties**(body: RID)

Restores the default inertia and center of mass of the body based on its shapes. This undoes any custom values previously set using body_set_param.

void **body_set_axis_velocity**(body: RID, axis_velocity: Vector2)

Modifies the body's linear velocity so that its projection to the axis `axis_velocity.normalized()` is exactly `axis_velocity.length()`. This is useful for jumping behavior.

void **body_set_collision_layer**(body: RID, layer: int)

Sets the physics layer or layers the body belongs to, via a bitmask.

void **body_set_collision_mask**(body: RID, mask: int)

Sets the physics layer or layers the body can collide with, via a bitmask.

void **body_set_collision_priority**(body: RID, priority: float)

Sets the body's collision priority. This is used in the depenetration phase of body_test_motion. The higher the priority is, the lower the penetration into the body will be.

void **body_set_constant_force**(body: RID, force: Vector2)

Sets the body's total constant positional force applied during each physics update.

See body_add_constant_force and body_add_constant_central_force.

---

void **body_set_constant_torque**(body: RID, torque: float)

Sets the body's total constant rotational force applied during each physics update.

See body_add_constant_torque.

---

void **body_set_continuous_collision_detection_mode**(body: RID, mode: CCDMode)

Sets the continuous collision detection mode using one of the CCDMode constants.

Continuous collision detection tries to predict where a moving body would collide in between physics updates, instead of moving it and correcting its movement if it collided.

---

void **body_set_force_integration_callback**(body: RID, callable: Callable, userdata: Variant = null)

Sets the body's custom force integration callback function to `callable`. Use an empty Callable (`Callable()`) to clear the custom callback.

The function `callable` will be called every physics tick, before the standard force integration (see body_set_omit_force_integration). It can be used for example to update the body's linear and angular velocity based on contact with other bodies.

If `userdata` is not `null`, the function `callable` must take the following two parameters:

1. `state`: a PhysicsDirectBodyState2D used to retrieve and modify the body's state,
2. `userdata`: a Variant; its value will be the `userdata` passed into this method.

If `userdata` is `null`, then `callable` must take only the `state` parameter.

---

void **body_set_max_contacts_reported**(body: RID, amount: int)

Sets the maximum number of contacts that the body can report. If `amount` is greater than zero, then the body will keep track of at most this many contacts with other bodies.

---

void **body_set_mode**(body: RID, mode: BodyMode)

Sets the body's mode. See BodyMode for the list of available modes.

---

void **body_set_omit_force_integration**(body: RID, enable: bool)

Sets whether the body omits the standard force integration. If `enable` is `true`, the body will not automatically use applied forces, torques, and damping to update the body's linear and angular velocity. In this case, body_set_force_integration_callback can be used to manually update the linear and angular velocity instead.

This method is called when the property RigidBody2D.custom_integrator is set.

---

void **body_set_param**(body: RID, param: BodyParameter, value: Variant)

Sets the value of the given body parameter. See BodyParameter for the list of available parameters.

---

void **body_set_shape**(body: RID, shape_idx: int, shape: RID)

Replaces the body's shape at the given index by another shape, while not affecting the `transform`, `disabled`, and one-way collision properties at the same index.

---

void **body_set_shape_as_one_way_collision**(body: RID, shape_idx: int, enable: bool, margin: float)

Sets the one-way collision properties of the body's shape with the given index. If `enable` is `true`, the one-way collision direction given by the shape's local upward axis `body_get_shape_transform(body, shape_idx).y` will be used to ignore collisions with the shape in the opposite direction, and to ensure depenetration of kinematic bodies happens in this direction.

---

void **body_set_shape_disabled**(body: RID, shape_idx: int, disabled: bool)

Sets the disabled property of the body's shape with the given index. If `disabled` is `true`, then the shape will be ignored in all collision detection.

---

void **body_set_shape_transform**(body: RID, shape_idx: int, transform: Transform2D) 🔗

Sets the local transform matrix of the body's shape with the given index.

---

void **body_set_space**(body: RID, space: RID) 🔗

Adds the body to the given space, after removing the body from the previously assigned space (if any). If the body's mode is set to BODY_MODE_RIGID, then adding the body to a space will have the following additional effects:

- If the parameter BODY_PARAM_CENTER_OF_MASS has never been set explicitly, then the value of that parameter will be recalculated based on the body's shapes.
- If the parameter BODY_PARAM_INERTIA is set to a value `<= 0.0`, then the value of that parameter will be recalculated based on the body's shapes, mass, and center of mass.

Note: To remove a body from a space without immediately adding it back elsewhere, use `PhysicsServer2D.body_set_space(body, RID())`.

---

void **body_set_state**(body: RID, state: BodyState, value: Variant) 🔗

Sets the value of a body's state. See BodyState for the list of available states.

Note: The state change doesn't take effect immediately. The state will change on the next physics frame.

---

void **body_set_state_sync_callback**(body: RID, callable: Callable) 🔗

Sets the body's state synchronization callback function to `callable`. Use an empty Callable (`Callable()`) to clear the callback.

The function `callable` will be called every physics frame, assuming that the body was active during the previous physics tick, and can be used to fetch the latest state from the physics server.

The function `callable` must take the following parameters:

1. `state`: a PhysicsDirectBodyState2D, used to retrieve the body's state.

---

bool **body_test_motion**(body: RID, parameters: PhysicsTestMotionParameters2D, result: PhysicsTestMotionResult2D = null)

Returns `true` if a collision would result from moving the body along a motion vector from a given point in space. See PhysicsTestMotionParameters2D for the available motion parameters. Optionally a PhysicsTestMotionResult2D object can be passed, which will be used to store the information about the resulting collision.

RID **capsule_shape_create**()

Creates a 2D capsule shape in the physics server, and returns the RID that identifies it. Use shape_set_data to set the capsule's height and radius.

RID **circle_shape_create**()

Creates a 2D circle shape in the physics server, and returns the RID that identifies it. Use shape_set_data to set the circle's radius.

RID **concave_polygon_shape_create**()

Creates a 2D concave polygon shape in the physics server, and returns the RID that identifies it. Use shape_set_data to set the concave polygon's segments.

RID **convex_polygon_shape_create**()

Creates a 2D convex polygon shape in the physics server, and returns the RID that identifies it. Use shape_set_data to set the convex polygon's points.

---

float **damped_spring_joint_get_param**(joint: RID, param: DampedSpringParam) const

Returns the value of the given damped spring joint parameter. See DampedSpringParam for the list of available parameters.

---

void **damped_spring_joint_set_param**(joint: RID, param: DampedSpringParam, value: float)

Sets the value of the given damped spring joint parameter. See DampedSpringParam for the list of available parameters.

---

void **free_rid**(rid: RID) 🔗

Destroys any of the objects created by PhysicsServer2D. If the RID passed is not one of the objects that can be created by PhysicsServer2D, an error will be printed to the console.

---

int **get_process_info**(process_info: ProcessInfo) 🔗

Returns information about the current state of the 2D physics engine. See ProcessInfo for the list of available states.

---

void **joint_clear**(joint: RID) 🔗

Destroys the joint with the given RID, creates a new uninitialized joint, and makes the RID refer to this new joint.

RID **joint_create**() 🔗

Creates a 2D joint in the physics server, and returns the RID that identifies it. To set the joint type, use joint_make_damped_spring, joint_make_groove or joint_make_pin. Use joint_set_param to set generic joint parameters.

---

void **joint_disable_collisions_between_bodies**(joint: RID, disable: bool) 🔗

Sets whether the bodies attached to the Joint2D will collide with each other.

---

float **joint_get_param**(joint: RID, param: JointParam) const 🔗

Returns the value of the given joint parameter. See JointParam for the list of available parameters.

JointType **joint_get_type**(joint: RID) const 🔗

Returns the joint's type (see JointType).

---

bool **joint_is_disabled_collisions_between_bodies**(joint: RID) const 🔗

Returns whether the bodies attached to the Joint2D will collide with each other.

---

void **joint_make_damped_spring**(joint: RID, anchor_a: Vector2, anchor_b: Vector2, body_a: RID, body_b: RID = RID()) 🔗

Makes the joint a damped spring joint, attached at the point `anchor_a` (given in global coordinates) on the body `body_a` and at the point `anchor_b` (given in global coordinates) on the body `body_b`. To set the parameters which are specific to the damped spring, see damped_spring_joint_set_param.

---

void **joint_make_groove**(joint: RID, groove1_a: Vector2, groove2_a: Vector2, anchor_b: Vector2, body_a: RID = RID(), body_b: RID = RID()) 🔗

Makes the joint a groove joint.

---

void **joint_make_pin**(joint: RID, anchor: Vector2, body_a: RID, body_b: RID = RID()) 🔗

Makes the joint a pin joint. If `body_b` is an empty RID, then `body_a` is pinned to the point `anchor` (given in global coordinates); otherwise, `body_a` is pinned to `body_b` at the point `anchor` (given in global coordinates). To set the parameters which are specific to the pin joint, see pin_joint_set_param.

---

void **joint_set_param**(joint: RID, param: JointParam, value: float) 🔗

Sets the value of the given joint parameter. See JointParam for the list of available parameters.

---

bool **pin_joint_get_flag**(joint: RID, flag: PinJointFlag) const 🔗

Gets a pin joint flag (see PinJointFlag constants).

---

float **pin_joint_get_param**(joint: RID, param: PinJointParam) const 🔗

Returns the value of a pin joint parameter. See PinJointParam for a list of available parameters.

---

void **pin_joint_set_flag**(joint: RID, flag: PinJointFlag, enabled: bool)

Sets a pin joint flag (see PinJointFlag constants).

---

void **pin_joint_set_param**(joint: RID, param: PinJointParam, value: float)

Sets a pin joint parameter. See PinJointParam for a list of available parameters.

RID **rectangle_shape_create**()

Creates a 2D rectangle shape in the physics server, and returns the RID that identifies it. Use shape_set_data to set the rectangle's half-extents.

RID **segment_shape_create**()

Creates a 2D segment shape in the physics server, and returns the RID that identifies it. Use shape_set_data to set the segment's start and end points.

RID **separation_ray_shape_create**()

Creates a 2D separation ray shape in the physics server, and returns the RID that identifies it. Use shape_set_data to set the shape's `length` and `slide_on_slope` properties.

---

void **set_active**(active: bool)

Activates or deactivates the 2D physics server. If `active` is `false`, then the physics server will not do anything in its physics step.

Variant **shape_get_data**(shape: RID) const

Returns the shape data that defines the configuration of the shape, such as the half-extents of a rectangle or the segments of a concave shape. See shape_set_data for the precise format of this data in each case.

ShapeType **shape_get_type**(shape: RID) const 🔗

Returns the shape's type (see ShapeType).

---

void **shape_set_data**(shape: RID, data: Variant) 🔗

Sets the shape data that defines the configuration of the shape. The `data` to be passed depends on the shape's type (see shape_get_type):

- SHAPE_WORLD_BOUNDARY: an array of length two containing a Vector2 `normal` direction and a float distance `d`,
- SHAPE_SEPARATION_RAY: a dictionary containing the key `length` with a float value and the key `slide_on_slope` with a bool value,
- SHAPE_SEGMENT: a Rect2 `rect` containing the first point of the segment in `rect.position` and the second point of the segment in `rect.size`,
- SHAPE_CIRCLE: a float `radius`,
- SHAPE_RECTANGLE: a Vector2 `half_extents`,
- SHAPE_CAPSULE: an array of length two (or a Vector2) containing a float `height` and a float `radius`,
- SHAPE_CONVEX_POLYGON: either a PackedVector2Array of points defining a convex polygon in counterclockwise order (the clockwise outward normal of each segment formed by consecutive points is calculated internally), or a PackedFloat32Array of length divisible by four so that every 4-tuple of floats contains the coordinates of a point followed by the coordinates of the clockwise outward normal vector to the segment between the current point and the next point,
- SHAPE_CONCAVE_POLYGON: a PackedVector2Array of length divisible by two (each pair of points forms one segment).

Warning: In the case of SHAPE_CONVEX_POLYGON, this method does not check if the points supplied actually form a convex polygon (unlike the CollisionPolygon2D.polygon property).

RID **space_create**() 🔗

Creates a 2D space in the physics server, and returns the RID that identifies it. A space contains bodies and areas, and controls the stepping of the physics simulation of the objects in it.

PhysicsDirectSpaceState2D **space_get_direct_state**(space: RID)

Returns the state of a space, a PhysicsDirectSpaceState2D. This object can be used for collision/intersection queries.

float **space_get_param**(space: RID, param: SpaceParameter) const

Returns the value of the given space parameter. See SpaceParameter for the list of available parameters.

bool **space_is_active**(space: RID) const

Returns `true` if the space is active.

void **space_set_active**(space: RID, active: bool)

Activates or deactivates the space. If `active` is `false`, then the physics server will not do anything with this space in its physics step.

void **space_set_param**(space: RID, param: SpaceParameter, value: float)

Sets the value of the given space parameter. See SpaceParameter for the list of available parameters.

RID **world_boundary_shape_create**()

Creates a 2D world boundary shape in the physics server, and returns the RID that identifies it. Use shape_set_data to set the shape's normal direction and distance properties.

PhysicsServer2DExtension

Inherits: PhysicsServer2D **<** Object

Provides virtual methods that can be overridden to create custom PhysicsServer2D implementations.

Description

This class extends PhysicsServer2D by providing additional virtual methods that can be overridden. When these methods are overridden, they will be called instead of the internal methods of the physics server.

Intended for use with GDExtension to create custom implementations of PhysicsServer2D.

Methods

void	_area_add_shape(area: RID, shape: RID, transform: Transform2D, disabled: bool) virtual
void	_area_attach_canvas_instance_id(area: RID, id: int) virtual
void	_area_attach_object_instance_id(area: RID, id: int) virtual
void	_area_clear_shapes(area: RID) virtual
RID	_area_create() virtual
int	_area_get_canvas_instance_id(area: RID) virtual const
int	_area_get_collision_layer(area: RID) virtual const

int	_area_get_collision_mask(area: RID) virtual const
int	_area_get_object_instance_id(area: RID) virtual const
Variant	_area_get_param(area: RID, param: AreaParameter) virtual const
RID	_area_get_shape(area: RID, shape_idx: int) virtual const
int	_area_get_shape_count(area: RID) virtual const
Transform2D	_area_get_shape_transform(area: RID, shape_idx: int) virtual const
RID	_area_get_space(area: RID) virtual const
Transform2D	_area_get_transform(area: RID) virtual const
void	_area_remove_shape(area: RID, shape_idx: int) virtual
void	_area_set_area_monitor_callback(area: RID, callback: Callable) virtual
void	_area_set_collision_layer(area: RID, layer: int) virtual
void	_area_set_collision_mask(area: RID, mask: int) virtual
void	_area_set_monitor_callback(area: RID, callback: Callable) virtual

void	_area_set_monitorable(area: RID, monitorable: bool) virtual
void	_area_set_param(area: RID, param: AreaParameter, value: Variant) virtual
void	_area_set_pickable(area: RID, pickable: bool) virtual
void	_area_set_shape(area: RID, shape_idx: int, shape: RID) virtual
void	_area_set_shape_disabled(area: RID, shape_idx: int, disabled: bool) virtual
void	_area_set_shape_transform(area: RID, shape_idx: int, transform: Transform2D) virtual
void	_area_set_space(area: RID, space: RID) virtual
void	_area_set_transform(area: RID, transform: Transform2D) virtual
void	_body_add_collision_exception(body: RID, excepted_body: RID) virtual
void	_body_add_constant_central_force(body: RID, force: Vector2) virtual
void	_body_add_constant_force(body: RID, force: Vector2, position: Vector2) virtual
void	_body_add_constant_torque(body: RID, torque: float) virtual

void	_body_add_shape(body: RID, shape: RID, transform: Transform2D, disabled: bool) virtual
void	_body_apply_central_force(body: RID, force: Vector2) virtual
void	_body_apply_central_impulse(body: RID, impulse: Vector2) virtual
void	_body_apply_force(body: RID, force: Vector2, position: Vector2) virtual
void	_body_apply_impulse(body: RID, impulse: Vector2, position: Vector2) virtual
void	_body_apply_torque(body: RID, torque: float) virtual
void	_body_apply_torque_impulse(body: RID, impulse: float) virtual
void	_body_attach_canvas_instance_id(body: RID, id: int) virtual
void	_body_attach_object_instance_id(body: RID, id: int) virtual
void	_body_clear_shapes(body: RID) virtual
bool	_body_collide_shape(body: RID, body_shape: int, shape: RID, shape_xform: Transform2D, motion: Vector2, results: `void*`, result_max: int, result_count: `int32_t*`) virtual
RID	_body_create() virtual

int	_body_get_canvas_instance_id(body: RID) virtual const
Array[RID]	_body_get_collision_exceptions(body: RID) virtual const
int	_body_get_collision_layer(body: RID) virtual const
int	_body_get_collision_mask(body: RID) virtual const
float	_body_get_collision_priority(body: RID) virtual const
Vector2	_body_get_constant_force(body: RID) virtual const
float	_body_get_constant_torque(body: RID) virtual const
float	_body_get_contacts_reported_depth_threshold(body: RID) virtual const
CCDMode	_body_get_continuous_collision_detection_mode(body: RID) virtual const
PhysicsDirectBodyState2D	_body_get_direct_state(body: RID) virtual
int	_body_get_max_contacts_reported(body: RID) virtual const
BodyMode	_body_get_mode(body: RID) virtual const
int	_body_get_object_instance_id(body: RID) virtual const

Variant	_body_get_param(body: RID, param: BodyParameter) virtual const
RID	_body_get_shape(body: RID, shape_idx: int) virtual const
int	_body_get_shape_count(body: RID) virtual const
Transform2D	_body_get_shape_transform(body: RID, shape_idx: int) virtual const
RID	_body_get_space(body: RID) virtual const
Variant	_body_get_state(body: RID, state: BodyState) virtual const
bool	_body_is_omitting_force_integration(body: RID) virtual const
void	_body_remove_collision_exception(body: RID, excepted_body: RID) virtual
void	_body_remove_shape(body: RID, shape_idx: int) virtual
void	_body_reset_mass_properties(body: RID) virtual
void	_body_set_axis_velocity(body: RID, axis_velocity: Vector2) virtual
void	_body_set_collision_layer(body: RID, layer: int) virtual
void	_body_set_collision_mask(body: RID, mask: int) virtual

void	_body_set_collision_priority(body: RID, priority: float) virtual
void	_body_set_constant_force(body: RID, force: Vector2) virtual
void	_body_set_constant_torque(body: RID, torque: float) virtual
void	_body_set_contacts_reported_depth_threshold(body: RID, threshold: float) virtual
void	_body_set_continuous_collision_detection_mode(body: RID, mode: CCDMode) virtual
void	_body_set_force_integration_callback(body: RID, callable: Callable, userdata: Variant) virtual
void	_body_set_max_contacts_reported(body: RID, amount: int) virtual
void	_body_set_mode(body: RID, mode: BodyMode) virtual
void	_body_set_omit_force_integration(body: RID, enable: bool) virtual
void	_body_set_param(body: RID, param: BodyParameter, value: Variant) virtual
void	_body_set_pickable(body: RID, pickable: bool) virtual
void	_body_set_shape(body: RID, shape_idx: int, shape:

	RID) virtual
void	_body_set_shape_as_one_way_collision(body: RID, shape_idx: int, enable: bool, margin: float) virtual
void	_body_set_shape_disabled(body: RID, shape_idx: int, disabled: bool) virtual
void	_body_set_shape_transform(body: RID, shape_idx: int, transform: Transform2D) virtual
void	_body_set_space(body: RID, space: RID) virtual
void	_body_set_state(body: RID, state: BodyState, value: Variant) virtual
void	_body_set_state_sync_callback(body: RID, callable: Callable) virtual
bool	_body_test_motion(body: RID, from: Transform2D, motion: Vector2, margin: float, collide_separation_ray: bool, recovery_as_collision: bool, result: `PhysicsServer2DExtensionMotionResult*`) virtual const
RID	_capsule_shape_create() virtual
RID	_circle_shape_create() virtual
RID	_concave_polygon_shape_create() virtual
RID	_convex_polygon_shape_create() virtual
float	_damped_spring_joint_get_param(joint: RID, param: DampedSpringParam) virtual const

void	_damped_spring_joint_set_param(joint: RID, param: DampedSpringParam, value: float) virtual
void	_end_sync() virtual
void	_finish() virtual
void	_flush_queries() virtual
void	_free_rid(rid: RID) virtual
int	_get_process_info(process_info: ProcessInfo) virtual
void	_init() virtual
bool	_is_flushing_queries() virtual const
void	_joint_clear(joint: RID) virtual
RID	_joint_create() virtual
void	_joint_disable_collisions_between_bodies(joint: RID, disable: bool) virtual
float	_joint_get_param(joint: RID, param: JointParam) virtual const
JointType	_joint_get_type(joint: RID) virtual const
bool	_joint_is_disabled_collisions_between_bodies(joint: RID) virtual const

void	_joint_make_damped_spring(joint: RID, anchor_a: Vector2, anchor_b: Vector2, body_a: RID, body_b: RID) virtual
void	_joint_make_groove(joint: RID, a_groove1: Vector2, a_groove2: Vector2, b_anchor: Vector2, body_a: RID, body_b: RID) virtual
void	_joint_make_pin(joint: RID, anchor: Vector2, body_a: RID, body_b: RID) virtual
void	_joint_set_param(joint: RID, param: JointParam, value: float) virtual
bool	_pin_joint_get_flag(joint: RID, flag: PinJointFlag) virtual const
float	_pin_joint_get_param(joint: RID, param: PinJointParam) virtual const
void	_pin_joint_set_flag(joint: RID, flag: PinJointFlag, enabled: bool) virtual
void	_pin_joint_set_param(joint: RID, param: PinJointParam, value: float) virtual
RID	_rectangle_shape_create() virtual
RID	_segment_shape_create() virtual
RID	_separation_ray_shape_create() virtual
void	_set_active(active: bool) virtual

bool	_shape_collide(shape_A: RID, xform_A: Transform2D, motion_A: Vector2, shape_B: RID, xform_B: Transform2D, motion_B: Vector2, results: `void*`, result_max: int, result_count: `int32_t*`) virtual
float	_shape_get_custom_solver_bias(shape: RID) virtual const
Variant	_shape_get_data(shape: RID) virtual const
ShapeType	_shape_get_type(shape: RID) virtual const
void	_shape_set_custom_solver_bias(shape: RID, bias: float) virtual
void	_shape_set_data(shape: RID, data: Variant) virtual
RID	_space_create() virtual
int	_space_get_contact_count(space: RID) virtual const
PackedVector2Array	_space_get_contacts(space: RID) virtual const
PhysicsDirectSpaceState2D	_space_get_direct_state(space: RID) virtual
float	_space_get_param(space: RID, param: SpaceParameter) virtual const
bool	_space_is_active(space: RID) virtual const
void	_space_set_active(space: RID, active: bool) virtual

void	_space_set_debug_contacts(space: RID, max_contacts: int) virtual
void	_space_set_param(space: RID, param: SpaceParameter, value: float) virtual
void	_step(step: float) virtual
void	_sync() virtual
RID	_world_boundary_shape_create() virtual
bool	body_test_motion_is_excluding_body(body: RID) const
bool	body_test_motion_is_excluding_object(object: int) const

Method Descriptions

void **_area_add_shape**(area: RID, shape: RID, transform: Transform2D, disabled: bool) virtual 🔗

Overridable version of PhysicsServer2D.area_add_shape.

void **_area_attach_canvas_instance_id**(area: RID, id: int) virtual 🔗

Overridable version of PhysicsServer2D.area_attach_canvas_instance_id.

void **_area_attach_object_instance_id**(area: RID, id: int) virtual 🔗

Overridable version of PhysicsServer2D.area_attach_object_instance_id.

void **_area_clear_shapes**(area: RID) virtual 🔗

Overridable version of PhysicsServer2D.area_clear_shapes.

RID **_area_create**() virtual

Overridable version of PhysicsServer2D.area_create.

---

int **_area_get_canvas_instance_id**(area: RID) virtual const

Overridable version of PhysicsServer2D.area_get_canvas_instance_id.

---

int **_area_get_collision_layer**(area: RID) virtual const

Overridable version of PhysicsServer2D.area_get_collision_layer.

---

int **_area_get_collision_mask**(area: RID) virtual const

Overridable version of PhysicsServer2D.area_get_collision_mask.

---

int **_area_get_object_instance_id**(area: RID) virtual const

Overridable version of PhysicsServer2D.area_get_object_instance_id.

Variant **_area_get_param**(area: RID, param: AreaParameter) virtual const

Overridable version of PhysicsServer2D.area_get_param.

RID **_area_get_shape**(area: RID, shape_idx: int) virtual const

Overridable version of PhysicsServer2D.area_get_shape.

---

int **_area_get_shape_count**(area: RID) virtual const

Overridable version of PhysicsServer2D.area_get_shape_count.

Transform2D **_area_get_shape_transform**(area: RID, shape_idx: int) virtual const

Overridable version of PhysicsServer2D.area_get_shape_transform.

RID **_area_get_space**(area: RID) virtual const

Overridable version of PhysicsServer2D.area_get_space.

Transform2D **_area_get_transform**(area: RID) virtual const

Overridable version of PhysicsServer2D.area_get_transform.

---

void **_area_remove_shape**(area: RID, shape_idx: int) virtual

Overridable version of PhysicsServer2D.area_remove_shape.

---

void **_area_set_area_monitor_callback**(area: RID, callback: Callable) virtual

Overridable version of PhysicsServer2D.area_set_area_monitor_callback.

---

void **_area_set_collision_layer**(area: RID, layer: int) virtual

Overridable version of PhysicsServer2D.area_set_collision_layer.

---

void **_area_set_collision_mask**(area: RID, mask: int) virtual

Overridable version of PhysicsServer2D.area_set_collision_mask.

---

void **_area_set_monitor_callback**(area: RID, callback: Callable) virtual

Overridable version of PhysicsServer2D.area_set_monitor_callback.

---

void **_area_set_monitorable**(area: RID, monitorable: bool) virtual

Overridable version of PhysicsServer2D.area_set_monitorable.

---

void **_area_set_param**(area: RID, param: AreaParameter, value: Variant) virtual

Overridable version of PhysicsServer2D.area_set_param.

---

void **_area_set_pickable**(area: RID, pickable: bool) virtual

If set to `true`, allows the area with the given RID to detect mouse inputs when the mouse cursor is hovering on it.

Overridable version of PhysicsServer2D's internal `area_set_pickable` method. Corresponds to CollisionObject2D.input_pickable.

---

void **_area_set_shape**(area: RID, shape_idx: int, shape: RID) virtual

Overridable version of PhysicsServer2D.area_set_shape.

---

void **_area_set_shape_disabled**(area: RID, shape_idx: int, disabled: bool) virtual

Overridable version of PhysicsServer2D.area_set_shape_disabled.

---

void **_area_set_shape_transform**(area: RID, shape_idx: int, transform: Transform2D) virtual

Overridable version of PhysicsServer2D.area_set_shape_transform.

---

void **_area_set_space**(area: RID, space: RID) virtual

Overridable version of PhysicsServer2D.area_set_space.

---

void **_area_set_transform**(area: RID, transform: Transform2D) virtual

Overridable version of PhysicsServer2D.area_set_transform.

---

void **_body_add_collision_exception**(body: RID, excepted_body: RID) virtual

Overridable version of PhysicsServer2D.body_add_collision_exception.

---

void **_body_add_constant_central_force**(body: RID, force: Vector2) virtual

Overridable version of PhysicsServer2D.body_add_constant_central_force.

---

void **_body_add_constant_force**(body: RID, force: Vector2, position: Vector2) virtual

Overridable version of PhysicsServer2D.body_add_constant_force.

---

void **_body_add_constant_torque**(body: RID, torque: float) virtual

Overridable version of PhysicsServer2D.body_add_constant_torque.

---

void **_body_add_shape**(body: RID, shape: RID, transform: Transform2D, disabled: bool) virtual

Overridable version of PhysicsServer2D.body_add_shape.

---

void **_body_apply_central_force**(body: RID, force: Vector2) virtual

Overridable version of PhysicsServer2D.body_apply_central_force.

---

void **_body_apply_central_impulse**(body: RID, impulse: Vector2) virtual

Overridable version of PhysicsServer2D.body_apply_central_impulse.

---

void **_body_apply_force**(body: RID, force: Vector2, position: Vector2) virtual

Overridable version of PhysicsServer2D.body_apply_force.

---

void **_body_apply_impulse**(body: RID, impulse: Vector2, position: Vector2) virtual

Overridable version of PhysicsServer2D.body_apply_impulse.

---

void **_body_apply_torque**(body: RID, torque: float) virtual

Overridable version of PhysicsServer2D.body_apply_torque.

---

void **_body_apply_torque_impulse**(body: RID, impulse: float) virtual

Overridable version of PhysicsServer2D.body_apply_torque_impulse.

---

void **_body_attach_canvas_instance_id**(body: RID, id: int) virtual

Overridable version of PhysicsServer2D.body_attach_canvas_instance_id.

---

void **_body_attach_object_instance_id**(body: RID, id: int) virtual

Overridable version of PhysicsServer2D.body_attach_object_instance_id.

---

void **_body_clear_shapes**(body: RID) virtual

Overridable version of PhysicsServer2D.body_clear_shapes.

---

bool **_body_collide_shape**(body: RID, body_shape: int, shape: RID, shape_xform: Transform2D, motion: Vector2, results: `void*`, result_max: int, result_count: `int32_t*`) virtual

Given a `body`, a `shape`, and their respective parameters, this method should return `true` if a collision between the two would occur, with additional details passed in `results`.

Overridable version of PhysicsServer2D's internal `shape_collide` method. Corresponds to PhysicsDirectSpaceState2D.collide_shape.

RID **_body_create**() virtual

Overridable version of PhysicsServer2D.body_create.

---

int **_body_get_canvas_instance_id**(body: RID) virtual const

Overridable version of PhysicsServer2D.body_get_canvas_instance_id.

Array[RID] **_body_get_collision_exceptions**(body: RID) virtual const 🔗

Returns the RIDs of all bodies added as collision exceptions for the given `body`. See also _body_add_collision_exception and _body_remove_collision_exception.

Overridable version of PhysicsServer2D's internal `body_get_collision_exceptions` method. Corresponds to PhysicsBody2D.get_collision_exceptions.

---

int **_body_get_collision_layer**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_collision_layer.

---

int **_body_get_collision_mask**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_collision_mask.

---

float **_body_get_collision_priority**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_collision_priority.

Vector2 **_body_get_constant_force**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_constant_force.

---

float **_body_get_constant_torque**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_constant_torque.

---

float **_body_get_contacts_reported_depth_threshold**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D's internal `body_get_contacts_reported_depth_threshold` method.

Note: This method is currently unused by Godot's default physics implementation.

CCDMode **_body_get_continuous_collision_detection_mode**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_continuous_collision_detection_mode.

PhysicsDirectBodyState2D **_body_get_direct_state**(body: RID) virtual 🔗

Overridable version of PhysicsServer2D.body_get_direct_state.

int **_body_get_max_contacts_reported**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_max_contacts_reported.

BodyMode **_body_get_mode**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_mode.

int **_body_get_object_instance_id**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_object_instance_id.

Variant **_body_get_param**(body: RID, param: BodyParameter) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_param.

RID **_body_get_shape**(body: RID, shape_idx: int) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_shape.

int **_body_get_shape_count**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_shape_count.

Transform2D **_body_get_shape_transform**(body: RID, shape_idx: int) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_shape_transform.

RID **_body_get_space**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_space.

Variant **_body_get_state**(body: RID, state: BodyState) virtual const 🔗

Overridable version of PhysicsServer2D.body_get_state.

bool **_body_is_omitting_force_integration**(body: RID) virtual const 🔗

Overridable version of PhysicsServer2D.body_is_omitting_force_integration.

void **_body_remove_collision_exception**(body: RID, excepted_body: RID) virtual 🔗

Overridable version of PhysicsServer2D.body_remove_collision_exception.

void **_body_remove_shape**(body: RID, shape_idx: int) virtual 🔗

Overridable version of PhysicsServer2D.body_remove_shape.

void **_body_reset_mass_properties**(body: RID) virtual 🔗

Overridable version of PhysicsServer2D.body_reset_mass_properties.

void **_body_set_axis_velocity**(body: RID, axis_velocity: Vector2) virtual 🔗

Overridable version of PhysicsServer2D.body_set_axis_velocity.

void **_body_set_collision_layer**(body: RID, layer: int) virtual 🔗

Overridable version of PhysicsServer2D.body_set_collision_layer.

void **_body_set_collision_mask**(body: RID, mask: int) virtual 🔗

Overridable version of PhysicsServer2D.body_set_collision_mask.

void **_body_set_collision_priority**(body: RID, priority: float) virtual

Overridable version of PhysicsServer2D.body_set_collision_priority.

void **_body_set_constant_force**(body: RID, force: Vector2) virtual

Overridable version of PhysicsServer2D.body_set_constant_force.

void **_body_set_constant_torque**(body: RID, torque: float) virtual

Overridable version of PhysicsServer2D.body_set_constant_torque.

void **_body_set_contacts_reported_depth_threshold**(body: RID, threshold: float) virtual

Overridable version of PhysicsServer2D's internal `body_set_contacts_reported_depth_threshold` method.

Note: This method is currently unused by Godot's default physics implementation.

void **_body_set_continuous_collision_detection_mode**(body: RID, mode: CCDMode) virtual

Overridable version of PhysicsServer2D.body_set_continuous_collision_detection_mode.

void **_body_set_force_integration_callback**(body: RID, callable: Callable, userdata: Variant) virtual

Overridable version of PhysicsServer2D.body_set_force_integration_callback.

void **_body_set_max_contacts_reported**(body: RID, amount: int) virtual

Overridable version of PhysicsServer2D.body_set_max_contacts_reported.

void **_body_set_mode**(body: RID, mode: BodyMode) virtual

Overridable version of PhysicsServer2D.body_set_mode.

void **_body_set_omit_force_integration**(body: RID, enable: bool) virtual

Overridable version of PhysicsServer2D.body_set_omit_force_integration.

void **_body_set_param**(body: RID, param: BodyParameter, value: Variant) virtual

Overridable version of PhysicsServer2D.body_set_param.

void **_body_set_pickable**(body: RID, pickable: bool) virtual

If set to `true`, allows the body with the given RID to detect mouse inputs when the mouse cursor is hovering on it.

Overridable version of PhysicsServer2D's internal `body_set_pickable` method. Corresponds to CollisionObject2D.input_pickable.

void **_body_set_shape**(body: RID, shape_idx: int, shape: RID) virtual

Overridable version of PhysicsServer2D.body_set_shape.

void **_body_set_shape_as_one_way_collision**(body: RID, shape_idx: int, enable: bool, margin: float) virtual

Overridable version of PhysicsServer2D.body_set_shape_as_one_way_collision.

void **_body_set_shape_disabled**(body: RID, shape_idx: int, disabled: bool) virtual

Overridable version of PhysicsServer2D.body_set_shape_disabled.

void **_body_set_shape_transform**(body: RID, shape_idx: int, transform: Transform2D) virtual

Overridable version of PhysicsServer2D.body_set_shape_transform.

void **_body_set_space**(body: RID, space: RID) virtual

Overridable version of PhysicsServer2D.body_set_space.

---

void **_body_set_state**(body: RID, state: BodyState, value: Variant) virtual 🔗

Overridable version of PhysicsServer2D.body_set_state.

---

void **_body_set_state_sync_callback**(body: RID, callable: Callable) virtual 🔗

Assigns the `body` to call the given `callable` during the synchronization phase of the loop, before _step is called. See also _sync.

Overridable version of PhysicsServer2D.body_set_state_sync_callback.

---

bool **_body_test_motion**(body: RID, from: Transform2D, motion: Vector2, margin: float, collide_separation_ray: bool, recovery_as_collision: bool, result: `PhysicsServer2DExtensionMotionResult*`) virtual const 🔗

Overridable version of PhysicsServer2D.body_test_motion. Unlike the exposed implementation, this method does not receive all of the arguments inside a PhysicsTestMotionParameters2D.

RID **_capsule_shape_create**() virtual 🔗

Overridable version of PhysicsServer2D.capsule_shape_create.

RID **_circle_shape_create**() virtual 🔗

Overridable version of PhysicsServer2D.circle_shape_create.

RID **_concave_polygon_shape_create**() virtual 🔗

Overridable version of PhysicsServer2D.concave_polygon_shape_create.

RID **_convex_polygon_shape_create**() virtual 🔗

Overridable version of PhysicsServer2D.convex_polygon_shape_create.

---

float **_damped_spring_joint_get_param**(joint: RID, param: DampedSpringParam) virtual const 🔗

Overridable version of PhysicsServer2D.damped_spring_joint_get_param.

---

void **_damped_spring_joint_set_param**(joint: RID, param: DampedSpringParam, value: float) virtual 🔗

Overridable version of PhysicsServer2D.damped_spring_joint_set_param.

---

void **_end_sync**() virtual 🔗

Called to indicate that the physics server has stopped synchronizing. It is in the loop's iteration/physics phase, and can access physics objects even if running on a separate thread. See also _sync.

Overridable version of PhysicsServer2D's internal `end_sync` method.

---

void **_finish**() virtual 🔗

Called when the main loop finalizes to shut down the physics server. See also MainLoop._finalize and _init.

Overridable version of PhysicsServer2D's internal `finish` method.

---

void **_flush_queries**() virtual 🔗

Called every physics step before _step to process all remaining queries.

Overridable version of PhysicsServer2D's internal `flush_queries` method.

---

void **_free_rid**(rid: RID) virtual 🔗

Overridable version of PhysicsServer2D.free_rid.

---

int **_get_process_info**(process_info: ProcessInfo) virtual 🔗

Overridable version of PhysicsServer2D.get_process_info.

---

void **_init**() virtual 🔗

Called when the main loop is initialized and creates a new instance of this physics server. See also MainLoop._initialize and _finish.

Overridable version of PhysicsServer2D's internal `init` method.

---

bool **_is_flushing_queries**() virtual const 🔗

Overridable method that should return `true` when the physics server is processing queries. See also _flush_queries.

Overridable version of PhysicsServer2D's internal `is_flushing_queries` method.

---

void **_joint_clear**(joint: RID) virtual 🔗

Overridable version of PhysicsServer2D.joint_clear.

RID **_joint_create**() virtual 🔗

Overridable version of PhysicsServer2D.joint_create.

---

void **_joint_disable_collisions_between_bodies**(joint: RID, disable: bool) virtual 🔗

Overridable version of PhysicsServer2D.joint_disable_collisions_between_bodies.

---

float **_joint_get_param**(joint: RID, param: JointParam) virtual const 🔗

Overridable version of PhysicsServer2D.joint_get_param.

JointType **_joint_get_type**(joint: RID) virtual const 🔗

Overridable version of PhysicsServer2D.joint_get_type.

---

bool **_joint_is_disabled_collisions_between_bodies**(joint: RID) virtual const 🔗

Overridable version of PhysicsServer2D.joint_is_disabled_collisions_between_bodies.

---

void **_joint_make_damped_spring**(joint: RID, anchor_a: Vector2, anchor_b: Vector2, body_a: RID, body_b: RID) virtual 🔗

Overridable version of PhysicsServer2D.joint_make_damped_spring.

---

void **_joint_make_groove**(joint: RID, a_groove1: Vector2, a_groove2: Vector2, b_anchor: Vector2, body_a: RID, body_b: RID) virtual 🔗

Overridable version of PhysicsServer2D.joint_make_groove.

---

void **_joint_make_pin**(joint: RID, anchor: Vector2, body_a: RID, body_b: RID) virtual 🔗

Overridable version of PhysicsServer2D.joint_make_pin.

---

void **_joint_set_param**(joint: RID, param: JointParam, value: float) virtual 🔗

Overridable version of PhysicsServer2D.joint_set_param.

---

bool **_pin_joint_get_flag**(joint: RID, flag: PinJointFlag) virtual const 🔗

Overridable version of PhysicsServer2D.pin_joint_get_flag.

---

float **_pin_joint_get_param**(joint: RID, param: PinJointParam) virtual const 🔗

Overridable version of PhysicsServer2D.pin_joint_get_param.

---

void **_pin_joint_set_flag**(joint: RID, flag: PinJointFlag, enabled: bool) virtual 🔗

Overridable version of PhysicsServer2D.pin_joint_set_flag.

---

void **_pin_joint_set_param**(joint: RID, param: PinJointParam, value: float) virtual 🔗

Overridable version of PhysicsServer2D.pin_joint_set_param.

RID **_rectangle_shape_create**() virtual 🔗

Overridable version of PhysicsServer2D.rectangle_shape_create.

RID **_segment_shape_create**() virtual 🔗

Overridable version of PhysicsServer2D.segment_shape_create.

RID **_separation_ray_shape_create**() virtual 🔗

Overridable version of PhysicsServer2D.separation_ray_shape_create.

void **_set_active**(active: bool) virtual 🔗

Overridable version of PhysicsServer2D.set_active.

bool **_shape_collide**(shape_A: RID, xform_A: Transform2D, motion_A: Vector2, shape_B: RID, xform_B: Transform2D, motion_B: Vector2, results: `void*`, result_max: int, result_count: `int32_t*`) virtual 🔗

Given two shapes and their parameters, should return `true` if a collision between the two would occur, with additional details passed in `results`.

Overridable version of PhysicsServer2D's internal `shape_collide` method. Corresponds to PhysicsDirectSpaceState2D.collide_shape.

float **_shape_get_custom_solver_bias**(shape: RID) virtual const 🔗

Should return the custom solver bias of the given `shape`, which defines how much bodies are forced to separate on contact when this shape is involved.

Overridable version of PhysicsServer2D's internal `shape_get_custom_solver_bias` method. Corresponds to Shape2D.custom_solver_bias.

Variant **_shape_get_data**(shape: RID) virtual const 🔗

Overridable version of PhysicsServer2D.shape_get_data.

ShapeType **_shape_get_type**(shape: RID) virtual const

Overridable version of PhysicsServer2D.shape_get_type.

---

void **_shape_set_custom_solver_bias**(shape: RID, bias: float) virtual

Should set the custom solver bias for the given `shape`. It defines how much bodies are forced to separate on contact.

Overridable version of PhysicsServer2D's internal `shape_get_custom_solver_bias` method. Corresponds to Shape2D.custom_solver_bias.

---

void **_shape_set_data**(shape: RID, data: Variant) virtual

Overridable version of PhysicsServer2D.shape_set_data.

RID **_space_create**() virtual

Overridable version of PhysicsServer2D.space_create.

---

int **_space_get_contact_count**(space: RID) virtual const

Should return how many contacts have occurred during the last physics step in the given `space`. See also _space_get_contacts and _space_set_debug_contacts.

Overridable version of PhysicsServer2D's internal `space_get_contact_count` method.

PackedVector2Array **_space_get_contacts**(space: RID) virtual const

Should return the positions of all contacts that have occurred during the last physics step in the given `space`. See also _space_get_contact_count and _space_set_debug_contacts.

Overridable version of PhysicsServer2D's internal `space_get_contacts` method.

PhysicsDirectSpaceState2D **_space_get_direct_state**(space: RID) virtual

Overridable version of PhysicsServer2D.space_get_direct_state.

float **_space_get_param**(space: RID, param: SpaceParameter) virtual const

Overridable version of PhysicsServer2D.space_get_param.

---

bool **_space_is_active**(space: RID) virtual const

Overridable version of PhysicsServer2D.space_is_active.

---

void **_space_set_active**(space: RID, active: bool) virtual

Overridable version of PhysicsServer2D.space_set_active.

---

void **_space_set_debug_contacts**(space: RID, max_contacts: int) virtual

Used internally to allow the given `space` to store contact points, up to `max_contacts`. This is automatically set for the main World2D's space when SceneTree.debug_collisions_hint is `true`, or by checking "Visible Collision Shapes" in the editor. Only works in debug builds.

Overridable version of PhysicsServer2D's internal `space_set_debug_contacts` method.

---

void **_space_set_param**(space: RID, param: SpaceParameter, value: float) virtual

Overridable version of PhysicsServer2D.space_set_param.

---

void **_step**(step: float) virtual

Called every physics step to process the physics simulation. `step` is the time elapsed since the last physics step, in seconds. It is usually the same as Node.get_physics_process_delta_time.

Overridable version of PhysicsServer2D's internal `step` method.

---

void **_sync**() virtual

Called to indicate that the physics server is synchronizing and cannot access physics states if running on a separate thread. See also _end_sync.

Overridable version of PhysicsServer2D's internal `sync` method.

RID **_world_boundary_shape_create**() virtual

Overridable version of PhysicsServer2D.world_boundary_shape_create.

---

bool **body_test_motion_is_excluding_body**(body: RID) const

Returns `true` if the body with the given RID is being excluded from _body_test_motion. See also Object.get_instance_id.

---

bool **body_test_motion_is_excluding_object**(object: int) const

Returns `true` if the object with the given instance ID is being excluded from _body_test_motion. See also Object.get_instance_id.

PhysicsServer2DManager

Inherits: Object

A singleton for managing PhysicsServer2D implementations.

Description

PhysicsServer2DManager is the API for registering PhysicsServer2D implementations and for setting the default implementation.

Note: It is not possible to switch physics servers at runtime. This class is only used on startup at the server initialization level, by Godot itself and possibly by GDExtensions.

Methods

void	register_server(name: String, create_callback: Callable)
void	set_default_server(name: String, priority: int)

Method Descriptions

void **register_server**(name: String, create_callback: Callable) 🔗

Register a PhysicsServer2D implementation by passing a `name` and a Callable that returns a PhysicsServer2D object.

---

void **set_default_server**(name: String, priority: int) 🔗

Set the default PhysicsServer2D implementation to the one identified by `name`, if `priority` is greater than the priority of the current default implementation.

PhysicsServer3D

Inherits: Object

Inherited By: PhysicsServer3DExtension

A server interface for low-level 3D physics access.

Description

PhysicsServer3D is the server responsible for all 3D physics. It can directly create and manipulate all physics objects:

- A *space* is a self-contained world for a physics simulation. It contains bodies, areas, and joints. Its state can be queried for collision and intersection information, and several parameters of the simulation can be modified.
- A *shape* is a geometric shape such as a sphere, a box, a cylinder, or a polygon. It can be used for collision detection by adding it to a body/area, possibly with an extra transformation relative to the body/area's origin. Bodies/areas can have multiple (transformed) shapes added to them, and a single shape can be added to bodies/areas multiple times with different local transformations.
- A *body* is a physical object which can be in static, kinematic, or rigid mode. Its state (such as position and velocity) can be queried and updated. A force integration callback can be set to customize the body's physics.
- An *area* is a region in space which can be used to detect bodies and areas entering and exiting it. A body monitoring callback can be set to report entering/exiting body shapes, and similarly an area monitoring callback can be set. Gravity and damping can be overridden within the area by setting area parameters.
- A *joint* is a constraint, either between two bodies or on one body relative to a point. Parameters such as the joint bias and the rest length of a spring joint can be adjusted.

Physics objects in **PhysicsServer3D** may be created and manipulated independently; they do not have to be tied to nodes in the scene tree.

Note: All the 3D physics nodes use the physics server internally. Adding a physics node to the scene tree will cause a corresponding physics object to be created in the physics server. A rigid body node registers a callback that updates the node's transform with the transform of the respective body object in the physics server (every physics update). An area node registers a callback to inform the area node about overlaps with the respective area object in the physics server. The raycast node queries the direct state of the relevant space in the physics server.

Methods

void	area_add_shape(area: RID, shape: RID, transform: Transform3D = Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0), disabled: bool = false)
void	area_attach_object_instance_id(area: RID, id: int)
void	area_clear_shapes(area: RID)
RID	area_create()
int	area_get_collision_layer(area: RID) const
int	area_get_collision_mask(area: RID) const
int	area_get_object_instance_id(area: RID) const
Variant	area_get_param(area: RID, param: AreaParameter) const
RID	area_get_shape(area: RID, shape_idx: int) const
int	area_get_shape_count(area: RID) const
Transform3D	area_get_shape_transform(area: RID, shape_idx: int) const
RID	area_get_space(area: RID) const

Transform3D	area_get_transform(area: RID) const
void	area_remove_shape(area: RID, shape_idx: int)
void	area_set_area_monitor_callback(area: RID, callback: Callable)
void	area_set_collision_layer(area: RID, layer: int)
void	area_set_collision_mask(area: RID, mask: int)
void	area_set_monitor_callback(area: RID, callback: Callable)
void	area_set_monitorable(area: RID, monitorable: bool)
void	area_set_param(area: RID, param: AreaParameter, value: Variant)
void	area_set_ray_pickable(area: RID, enable: bool)
void	area_set_shape(area: RID, shape_idx: int, shape: RID)
void	area_set_shape_disabled(area: RID, shape_idx: int, disabled: bool)
void	area_set_shape_transform(area: RID, shape_idx: int, transform: Transform3D)
void	area_set_space(area: RID, space: RID)

void	area_set_transform(area: RID, transform: Transform3D)
void	body_add_collision_exception(body: RID, excepted_body: RID)
void	body_add_constant_central_force(body: RID, force: Vector3)
void	body_add_constant_force(body: RID, force: Vector3, position: Vector3 = Vector3(0, 0, 0))
void	body_add_constant_torque(body: RID, torque: Vector3)
void	body_add_shape(body: RID, shape: RID, transform: Transform3D = Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0), disabled: bool = false)
void	body_apply_central_force(body: RID, force: Vector3)
void	body_apply_central_impulse(body: RID, impulse: Vector3)
void	body_apply_force(body: RID, force: Vector3, position: Vector3 = Vector3(0, 0, 0))
void	body_apply_impulse(body: RID, impulse: Vector3, position: Vector3 = Vector3(0, 0, 0))
void	body_apply_torque(body: RID, torque: Vector3)
void	body_apply_torque_impulse(body: RID, impulse: Vector3)

void	body_attach_object_instance_id(body: RID, id: int)
void	body_clear_shapes(body: RID)
RID	body_create()
int	body_get_collision_layer(body: RID) const
int	body_get_collision_mask(body: RID) const
float	body_get_collision_priority(body: RID) const
Vector3	body_get_constant_force(body: RID) const
Vector3	body_get_constant_torque(body: RID) const
PhysicsDirectBodyState3D	body_get_direct_state(body: RID)
int	body_get_max_contacts_reported(body: RID) const
BodyMode	body_get_mode(body: RID) const
int	body_get_object_instance_id(body: RID) const
Variant	body_get_param(body: RID, param: BodyParameter) const
RID	body_get_shape(body: RID, shape_idx: int) const

int	body_get_shape_count(body: RID) const
Transform3D	body_get_shape_transform(body: RID, shape_idx: int) const
RID	body_get_space(body: RID) const
Variant	body_get_state(body: RID, state: BodyState) const
bool	body_is_axis_locked(body: RID, axis: BodyAxis) const
bool	body_is_continuous_collision_detection_enabled(body: RID) const
bool	body_is_omitting_force_integration(body: RID) const
void	body_remove_collision_exception(body: RID, excepted_body: RID)
void	body_remove_shape(body: RID, shape_idx: int)
void	body_reset_mass_properties(body: RID)
void	body_set_axis_lock(body: RID, axis: BodyAxis, lock: bool)
void	body_set_axis_velocity(body: RID, axis_velocity: Vector3)
void	body_set_collision_layer(body: RID, layer: int)
void	body_set_collision_mask(body: RID, mask: int)

void	body_set_collision_priority(body: RID, priority: float)
void	body_set_constant_force(body: RID, force: Vector3)
void	body_set_constant_torque(body: RID, torque: Vector3)
void	body_set_enable_continuous_collision_detection(body: RID, enable: bool)
void	body_set_force_integration_callback(body: RID, callable: Callable, userdata: Variant = null)
void	body_set_max_contacts_reported(body: RID, amount: int)
void	body_set_mode(body: RID, mode: BodyMode)
void	body_set_omit_force_integration(body: RID, enable: bool)
void	body_set_param(body: RID, param: BodyParameter, value: Variant)
void	body_set_ray_pickable(body: RID, enable: bool)
void	body_set_shape(body: RID, shape_idx: int, shape: RID)
void	body_set_shape_disabled(body: RID, shape_idx: int, disabled: bool)

void	body_set_shape_transform(body: RID, shape_idx: int, transform: Transform3D)
void	body_set_space(body: RID, space: RID)
void	body_set_state(body: RID, state: BodyState, value: Variant)
void	body_set_state_sync_callback(body: RID, callable: Callable)
bool	body_test_motion(body: RID, parameters: PhysicsTestMotionParameters3D, result: PhysicsTestMotionResult3D = null)
RID	box_shape_create()
RID	capsule_shape_create()
RID	concave_polygon_shape_create()
float	cone_twist_joint_get_param(joint: RID, param: ConeTwistJointParam) const
void	cone_twist_joint_set_param(joint: RID, param: ConeTwistJointParam, value: float)
RID	convex_polygon_shape_create()
RID	custom_shape_create()
RID	cylinder_shape_create()

void	free_rid(rid: RID)
bool	generic_6dof_joint_get_flag(joint: RID, axis: Vector3.Axis, flag: G6DOFJointAxisFlag) const
float	generic_6dof_joint_get_param(joint: RID, axis: Vector3.Axis, param: G6DOFJointAxisParam) const
void	generic_6dof_joint_set_flag(joint: RID, axis: Vector3.Axis, flag: G6DOFJointAxisFlag, enable: bool)
void	generic_6dof_joint_set_param(joint: RID, axis: Vector3.Axis, param: G6DOFJointAxisParam, value: float)
int	get_process_info(process_info: ProcessInfo)
RID	heightmap_shape_create()
bool	hinge_joint_get_flag(joint: RID, flag: HingeJointFlag) const
float	hinge_joint_get_param(joint: RID, param: HingeJointParam) const
void	hinge_joint_set_flag(joint: RID, flag: HingeJointFlag, enabled: bool)
void	hinge_joint_set_param(joint: RID, param: HingeJointParam, value: float)
void	joint_clear(joint: RID)

RID	joint_create()
void	joint_disable_collisions_between_bodies(joint: RID, disable: bool)
int	joint_get_solver_priority(joint: RID) const
JointType	joint_get_type(joint: RID) const
bool	joint_is_disabled_collisions_between_bodies(joint: RID) const
void	joint_make_cone_twist(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D)
void	joint_make_generic_6dof(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D)
void	joint_make_hinge(joint: RID, body_A: RID, hinge_A: Transform3D, body_B: RID, hinge_B: Transform3D)
void	joint_make_pin(joint: RID, body_A: RID, local_A: Vector3, body_B: RID, local_B: Vector3)
void	joint_make_slider(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D)
void	joint_set_solver_priority(joint: RID, priority: int)
Vector3	pin_joint_get_local_a(joint: RID) const

Vector3	pin_joint_get_local_b(joint: RID) const
float	pin_joint_get_param(joint: RID, param: PinJointParam) const
void	pin_joint_set_local_a(joint: RID, local_A: Vector3)
void	pin_joint_set_local_b(joint: RID, local_B: Vector3)
void	pin_joint_set_param(joint: RID, param: PinJointParam, value: float)
RID	separation_ray_shape_create()
void	set_active(active: bool)
Variant	shape_get_data(shape: RID) const
float	shape_get_margin(shape: RID) const
ShapeType	shape_get_type(shape: RID) const
void	shape_set_data(shape: RID, data: Variant)
void	shape_set_margin(shape: RID, margin: float)
float	slider_joint_get_param(joint: RID, param: SliderJointParam) const
void	slider_joint_set_param(joint: RID, param: SliderJointParam, value: float)

void	soft_body_add_collision_exception(body: RID, body_b: RID)
RID	soft_body_create()
AABB	soft_body_get_bounds(body: RID) const
int	soft_body_get_collision_layer(body: RID) const
int	soft_body_get_collision_mask(body: RID) const
float	soft_body_get_damping_coefficient(body: RID) const
float	soft_body_get_drag_coefficient(body: RID) const
float	soft_body_get_linear_stiffness(body: RID) const
Vector3	soft_body_get_point_global_position(body: RID, point_index: int) const
float	soft_body_get_pressure_coefficient(body: RID) const
int	soft_body_get_simulation_precision(body: RID) const
RID	soft_body_get_space(body: RID) const
Variant	soft_body_get_state(body: RID, state: BodyState) const
float	soft_body_get_total_mass(body: RID) const

bool	soft_body_is_point_pinned(body: RID, point_index: int) const
void	soft_body_move_point(body: RID, point_index: int, global_position: Vector3)
void	soft_body_pin_point(body: RID, point_index: int, pin: bool)
void	soft_body_remove_all_pinned_points(body: RID)
void	soft_body_remove_collision_exception(body: RID, body_b: RID)
void	soft_body_set_collision_layer(body: RID, layer: int)
void	soft_body_set_collision_mask(body: RID, mask: int)
void	soft_body_set_damping_coefficient(body: RID, damping_coefficient: float)
void	soft_body_set_drag_coefficient(body: RID, drag_coefficient: float)
void	soft_body_set_linear_stiffness(body: RID, stiffness: float)
void	soft_body_set_mesh(body: RID, mesh: RID)
void	soft_body_set_pressure_coefficient(body: RID, pressure_coefficient: float)
void	soft_body_set_ray_pickable(body: RID, enable: bool)

void	soft_body_set_simulation_precision(body: RID, simulation_precision: int)
void	soft_body_set_space(body: RID, space: RID)
void	soft_body_set_state(body: RID, state: BodyState, variant: Variant)
void	soft_body_set_total_mass(body: RID, total_mass: float)
void	soft_body_set_transform(body: RID, transform: Transform3D)
void	soft_body_update_rendering_server(body: RID, rendering_server_handler: PhysicsServer3DRenderingServerHandler)
RID	space_create()
PhysicsDirectSpaceState3D	space_get_direct_state(space: RID)
float	space_get_param(space: RID, param: SpaceParameter) const
bool	space_is_active(space: RID) const
void	space_set_active(space: RID, active: bool)
void	space_set_param(space: RID, param: SpaceParameter, value: float)

RID	sphere_shape_create()
RID	world_boundary_shape_create()

Enumerations

enum **JointType**: 🔗

JointType **JOINT_TYPE_PIN** = `0`

The Joint3D is a PinJoint3D.

JointType **JOINT_TYPE_HINGE** = `1`

The Joint3D is a HingeJoint3D.

JointType **JOINT_TYPE_SLIDER** = `2`

The Joint3D is a SliderJoint3D.

JointType **JOINT_TYPE_CONE_TWIST** = `3`

The Joint3D is a ConeTwistJoint3D.

JointType **JOINT_TYPE_6DOF** = `4`

The Joint3D is a Generic6DOFJoint3D.

JointType **JOINT_TYPE_MAX** = `5`

Represents the size of the JointType enum.

---

enum **PinJointParam**: 🔗

PinJointParam **PIN_JOINT_BIAS** = `0`

The strength with which the pinned objects try to stay in positional relation to each other.

The higher, the stronger.

PinJointParam **PIN_JOINT_DAMPING** = `1`

The strength with which the pinned objects try to stay in velocity relation to each other.

The higher, the stronger.

PinJointParam **PIN_JOINT_IMPULSE_CLAMP** = `2`

If above 0, this value is the maximum value for an impulse that this Joint3D puts on its ends.

enum **HingeJointParam**: 🔗

HingeJointParam **HINGE_JOINT_BIAS** = `0`

The speed with which the two bodies get pulled together when they move in different directions.

HingeJointParam **HINGE_JOINT_LIMIT_UPPER** = `1`

The maximum rotation across the Hinge.

HingeJointParam **HINGE_JOINT_LIMIT_LOWER** = `2`

The minimum rotation across the Hinge.

HingeJointParam **HINGE_JOINT_LIMIT_BIAS** = `3`

The speed with which the rotation across the axis perpendicular to the hinge gets corrected.

HingeJointParam **HINGE_JOINT_LIMIT_SOFTNESS** = `4`

There is currently no description for this enum. Please help us by contributing one!

HingeJointParam **HINGE_JOINT_LIMIT_RELAXATION** = `5`

The lower this value, the more the rotation gets slowed down.

HingeJointParam **HINGE_JOINT_MOTOR_TARGET_VELOCITY** = `6`

Target speed for the motor.

HingeJointParam **HINGE_JOINT_MOTOR_MAX_IMPULSE** = `7`

Maximum acceleration for the motor.

enum **HingeJointFlag**:

HingeJointFlag **HINGE_JOINT_FLAG_USE_LIMIT** = `0`

If `true`, the Hinge has a maximum and a minimum rotation.

HingeJointFlag **HINGE_JOINT_FLAG_ENABLE_MOTOR** = `1`

If `true`, a motor turns the Hinge.

enum **SliderJointParam**:

SliderJointParam **SLIDER_JOINT_LINEAR_LIMIT_UPPER** = `0`

The maximum difference between the pivot points on their X axis before damping happens.

SliderJointParam **SLIDER_JOINT_LINEAR_LIMIT_LOWER** = `1`

The minimum difference between the pivot points on their X axis before damping happens.

SliderJointParam **SLIDER_JOINT_LINEAR_LIMIT_SOFTNESS** = `2`

A factor applied to the movement across the slider axis once the limits get surpassed. The lower, the slower the movement.

SliderJointParam **SLIDER_JOINT_LINEAR_LIMIT_RESTITUTION** = `3`

The amount of restitution once the limits are surpassed. The lower, the more velocity-energy gets lost.

SliderJointParam **SLIDER_JOINT_LINEAR_LIMIT_DAMPING** = `4`

The amount of damping once the slider limits are surpassed.

SliderJointParam **SLIDER_JOINT_LINEAR_MOTION_SOFTNESS** = `5`

A factor applied to the movement across the slider axis as long as the slider is in the limits. The lower, the slower the movement.

SliderJointParam **SLIDER_JOINT_LINEAR_MOTION_RESTITUTION** = `6`

The amount of restitution inside the slider limits.

SliderJointParam **SLIDER_JOINT_LINEAR_MOTION_DAMPING** = `7`

The amount of damping inside the slider limits.

SliderJointParam **SLIDER_JOINT_LINEAR_ORTHOGONAL_SOFTNESS** = `8`

A factor applied to the movement across axes orthogonal to the slider.

SliderJointParam **SLIDER_JOINT_LINEAR_ORTHOGONAL_RESTITUTION** = `9`

The amount of restitution when movement is across axes orthogonal to the slider.

SliderJointParam **SLIDER_JOINT_LINEAR_ORTHOGONAL_DAMPING** = `10`

The amount of damping when movement is across axes orthogonal to the slider.

SliderJointParam **SLIDER_JOINT_ANGULAR_LIMIT_UPPER** = `11`

The upper limit of rotation in the slider.

SliderJointParam **SLIDER_JOINT_ANGULAR_LIMIT_LOWER** = `12`

The lower limit of rotation in the slider.

SliderJointParam **SLIDER_JOINT_ANGULAR_LIMIT_SOFTNESS** = `13`

A factor applied to the all rotation once the limit is surpassed.

SliderJointParam **SLIDER_JOINT_ANGULAR_LIMIT_RESTITUTION** = `14`

The amount of restitution of the rotation when the limit is surpassed.

SliderJointParam **SLIDER_JOINT_ANGULAR_LIMIT_DAMPING** = `15`

The amount of damping of the rotation when the limit is surpassed.

SliderJointParam **SLIDER_JOINT_ANGULAR_MOTION_SOFTNESS** = `16`

A factor that gets applied to the all rotation in the limits.

SliderJointParam **SLIDER_JOINT_ANGULAR_MOTION_RESTITUTION** = `17`

The amount of restitution of the rotation in the limits.

SliderJointParam **SLIDER_JOINT_ANGULAR_MOTION_DAMPING** = `18`

The amount of damping of the rotation in the limits.

SliderJointParam **SLIDER_JOINT_ANGULAR_ORTHOGONAL_SOFTNESS** = `19`

A factor that gets applied to the all rotation across axes orthogonal to the slider.

SliderJointParam **SLIDER_JOINT_ANGULAR_ORTHOGONAL_RESTITUTION** = `20`

The amount of restitution of the rotation across axes orthogonal to the slider.

SliderJointParam **SLIDER_JOINT_ANGULAR_ORTHOGONAL_DAMPING** = `21`

The amount of damping of the rotation across axes orthogonal to the slider.

SliderJointParam **SLIDER_JOINT_MAX** = `22`

Represents the size of the SliderJointParam enum.

---

enum **ConeTwistJointParam**: 🔗

ConeTwistJointParam **CONE_TWIST_JOINT_SWING_SPAN** = `0`

Swing is rotation from side to side, around the axis perpendicular to the twist axis.

The swing span defines, how much rotation will not get corrected along the swing axis.

Could be defined as looseness in the ConeTwistJoint3D.

If below 0.05, this behavior is locked.

ConeTwistJointParam **CONE_TWIST_JOINT_TWIST_SPAN** = `1`

Twist is the rotation around the twist axis, this value defined how far the joint can twist.

Twist is locked if below 0.05.

ConeTwistJointParam **CONE_TWIST_JOINT_BIAS** = `2`

The speed with which the swing or twist will take place.

The higher, the faster.

ConeTwistJointParam **CONE_TWIST_JOINT_SOFTNESS** = `3`

The ease with which the Joint3D twists, if it's too low, it takes more force to twist the joint.

ConeTwistJointParam **CONE_TWIST_JOINT_RELAXATION** = `4`

Defines, how fast the swing- and twist-speed-difference on both sides gets synced.

---

enum **G6DOFJointAxisParam**: 🔗

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_LOWER_LIMIT** = `0`

The minimum difference between the pivot points' axes.

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_UPPER_LIMIT** = `1`

The maximum difference between the pivot points' axes.

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_LIMIT_SOFTNESS** = `2`

A factor that gets applied to the movement across the axes. The lower, the slower the movement.

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_RESTITUTION** = `3`

The amount of restitution on the axes movement. The lower, the more velocity-energy gets lost.

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_DAMPING** = `4`

The amount of damping that happens at the linear motion across the axes.

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_MOTOR_TARGET_VELOCITY** = `5`

The velocity that the joint's linear motor will attempt to reach.

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_MOTOR_FORCE_LIMIT** = `6`

The maximum force that the linear motor can apply while trying to reach the target velocity.

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_SPRING_STIFFNESS** = `7`

There is currently no description for this enum. Please help us by contributing one!

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_SPRING_DAMPING** = `8`

There is currently no description for this enum. Please help us by contributing one!

G6DOFJointAxisParam **G6DOF_JOINT_LINEAR_SPRING_EQUILIBRIUM_POINT** = `9`

There is currently no description for this enum. Please help us by contributing one!

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_LOWER_LIMIT** = `10`

The minimum rotation in negative direction to break loose and rotate around the axes.

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_UPPER_LIMIT** = `11`

The minimum rotation in positive direction to break loose and rotate around the axes.

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_LIMIT_SOFTNESS** = `12`

A factor that gets multiplied onto all rotations across the axes.

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_DAMPING** = `13`

The amount of rotational damping across the axes. The lower, the more damping occurs.

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_RESTITUTION** = `14`

The amount of rotational restitution across the axes. The lower, the more restitution occurs.

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_FORCE_LIMIT** = `15`

The maximum amount of force that can occur, when rotating around the axes.

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_ERP** = `16`

When correcting the crossing of limits in rotation across the axes, this error tolerance factor defines how much the correction gets slowed down. The lower, the slower.

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_MOTOR_TARGET_VELOCITY** = `17`

Target speed for the motor at the axes.

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_MOTOR_FORCE_LIMIT** = `18`

Maximum acceleration for the motor at the axes.

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_SPRING_STIFFNESS** = `19`

There is currently no description for this enum. Please help us by contributing one!

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_SPRING_DAMPING** = `20`

There is currently no description for this enum. Please help us by contributing one!

G6DOFJointAxisParam **G6DOF_JOINT_ANGULAR_SPRING_EQUILIBRIUM_POINT** = `21`

There is currently no description for this enum. Please help us by contributing one!

G6DOFJointAxisParam **G6DOF_JOINT_MAX** = `22`

Represents the size of the G6DOFJointAxisParam enum.

---

enum **G6DOFJointAxisFlag**: 🔗

G6DOFJointAxisFlag **G6DOF_JOINT_FLAG_ENABLE_LINEAR_LIMIT** = `0`

If set, linear motion is possible within the given limits.

G6DOFJointAxisFlag **G6DOF_JOINT_FLAG_ENABLE_ANGULAR_LIMIT** = `1`

If set, rotational motion is possible.

G6DOFJointAxisFlag **G6DOF_JOINT_FLAG_ENABLE_ANGULAR_SPRING** = `2`

There is currently no description for this enum. Please help us by contributing one!

G6DOFJointAxisFlag **G6DOF_JOINT_FLAG_ENABLE_LINEAR_SPRING** = `3`

There is currently no description for this enum. Please help us by contributing one!

G6DOFJointAxisFlag **G6DOF_JOINT_FLAG_ENABLE_MOTOR** = `4`

If set, there is a rotational motor across these axes.

G6DOFJointAxisFlag **G6DOF_JOINT_FLAG_ENABLE_LINEAR_MOTOR** = `5`

If set, there is a linear motor on this axis that targets a specific velocity.

G6DOFJointAxisFlag **G6DOF_JOINT_FLAG_MAX** = `6`

Represents the size of the G6DOFJointAxisFlag enum.

---

enum **ShapeType**: 🔗

ShapeType **SHAPE_WORLD_BOUNDARY** = `0`

The Shape3D is a WorldBoundaryShape3D.

ShapeType **SHAPE_SEPARATION_RAY** = `1`

The Shape3D is a SeparationRayShape3D.

ShapeType **SHAPE_SPHERE** = `2`

The Shape3D is a SphereShape3D.

ShapeType **SHAPE_BOX** = `3`

The Shape3D is a BoxShape3D.

ShapeType **SHAPE_CAPSULE** = `4`

The Shape3D is a CapsuleShape3D.

ShapeType **SHAPE_CYLINDER** = `5`

The Shape3D is a CylinderShape3D.

ShapeType **SHAPE_CONVEX_POLYGON** = `6`

The Shape3D is a ConvexPolygonShape3D.

ShapeType **SHAPE_CONCAVE_POLYGON** = `7`

The Shape3D is a ConcavePolygonShape3D.

ShapeType **SHAPE_HEIGHTMAP** = `8`

The Shape3D is a HeightMapShape3D.

ShapeType **SHAPE_SOFT_BODY** = `9`

The Shape3D is used internally for a soft body. Any attempt to create this kind of shape results in an error.

ShapeType **SHAPE_CUSTOM** = `10`

This constant is used internally by the engine. Any attempt to create this kind of shape results in an error.

enum **AreaParameter**: 🔗

AreaParameter **AREA_PARAM_GRAVITY_OVERRIDE_MODE** = `0`

Constant to set/get gravity override mode in an area. See AreaSpaceOverrideMode for possible values.

AreaParameter **AREA_PARAM_GRAVITY** = `1`

Constant to set/get gravity strength in an area.

AreaParameter **AREA_PARAM_GRAVITY_VECTOR** = `2`

Constant to set/get gravity vector/center in an area.

AreaParameter **AREA_PARAM_GRAVITY_IS_POINT** = `3`

Constant to set/get whether the gravity vector of an area is a direction, or a center point.

AreaParameter **AREA_PARAM_GRAVITY_POINT_UNIT_DISTANCE** = `4`

Constant to set/get the distance at which the gravity strength is equal to the gravity controlled by AREA_PARAM_GRAVITY. For example, on a planet 100 meters in radius with a surface gravity of 4.0 m/s², set the gravity to 4.0 and the unit distance to 100.0. The gravity will have falloff according to the inverse square law, so in the example, at 200 meters from the center the gravity will be 1.0 m/s² (twice the distance, 1/4th the gravity), at 50 meters it will be 16.0 m/s² (half the distance, 4x the gravity), and so on.

The above is true only when the unit distance is a positive number. When this is set to 0.0, the gravity will be constant regardless of distance.

AreaParameter **AREA_PARAM_LINEAR_DAMP_OVERRIDE_MODE** = `5`

Constant to set/get linear damping override mode in an area. See AreaSpaceOverrideMode for possible values.

AreaParameter **AREA_PARAM_LINEAR_DAMP** = `6`

Constant to set/get the linear damping factor of an area.

AreaParameter **AREA_PARAM_ANGULAR_DAMP_OVERRIDE_MODE** = `7`

Constant to set/get angular damping override mode in an area. See AreaSpaceOverrideMode for possible values.

AreaParameter **AREA_PARAM_ANGULAR_DAMP** = `8`

Constant to set/get the angular damping factor of an area.

AreaParameter **AREA_PARAM_PRIORITY** = `9`

Constant to set/get the priority (order of processing) of an area.

AreaParameter **AREA_PARAM_WIND_FORCE_MAGNITUDE** = `10`

Constant to set/get the magnitude of area-specific wind force. This wind force only applies to SoftBody3D nodes. Other physics bodies are currently not affected by wind.

AreaParameter **AREA_PARAM_WIND_SOURCE** = `11`

Constant to set/get the 3D vector that specifies the origin from which an area-specific wind blows.

AreaParameter **AREA_PARAM_WIND_DIRECTION** = `12`

Constant to set/get the 3D vector that specifies the direction in which an area-specific wind blows.

AreaParameter **AREA_PARAM_WIND_ATTENUATION_FACTOR** = `13`

Constant to set/get the exponential rate at which wind force decreases with distance from its origin.

---

enum **AreaSpaceOverrideMode**: 🔗

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_DISABLED** = `0`

This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_COMBINE** = `1`

This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_COMBINE_REPLACE** = `2`

This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_REPLACE** = `3`

This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.

AreaSpaceOverrideMode **AREA_SPACE_OVERRIDE_REPLACE_COMBINE** = `4`

This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.

enum **BodyMode**: 🔗

BodyMode **BODY_MODE_STATIC** = `0`

Constant for static bodies. In this mode, a body can be only moved by user code and doesn't collide with other bodies along its path when moved.

BodyMode **BODY_MODE_KINEMATIC** = `1`

Constant for kinematic bodies. In this mode, a body can be only moved by user code and collides with other bodies along its path.

BodyMode **BODY_MODE_RIGID** = `2`

Constant for rigid bodies. In this mode, a body can be pushed by other bodies and has forces applied.

BodyMode **BODY_MODE_RIGID_LINEAR** = `3`

Constant for linear rigid bodies. In this mode, a body can not rotate, and only its linear velocity is affected by external forces.

---

enum **BodyParameter**: 🔗

BodyParameter **BODY_PARAM_BOUNCE** = `0`

Constant to set/get a body's bounce factor.

BodyParameter **BODY_PARAM_FRICTION** = `1`

Constant to set/get a body's friction.

BodyParameter **BODY_PARAM_MASS** = `2`

Constant to set/get a body's mass.

BodyParameter **BODY_PARAM_INERTIA** = `3`

Constant to set/get a body's inertia.

BodyParameter **BODY_PARAM_CENTER_OF_MASS** = `4`

Constant to set/get a body's center of mass position in the body's local coordinate system.

BodyParameter **BODY_PARAM_GRAVITY_SCALE** = `5`

Constant to set/get a body's gravity multiplier.

BodyParameter **BODY_PARAM_LINEAR_DAMP_MODE** = `6`

Constant to set/get a body's linear damping mode. See BodyDampMode for possible values.

BodyParameter **BODY_PARAM_ANGULAR_DAMP_MODE** = `7`

Constant to set/get a body's angular damping mode. See BodyDampMode for possible values.

BodyParameter **BODY_PARAM_LINEAR_DAMP** = `8`

Constant to set/get a body's linear damping factor.

BodyParameter **BODY_PARAM_ANGULAR_DAMP** = `9`

Constant to set/get a body's angular damping factor.

BodyParameter **BODY_PARAM_MAX** = `10`

Represents the size of the BodyParameter enum.

enum **BodyDampMode**: 🔗

BodyDampMode **BODY_DAMP_MODE_COMBINE** = `0`

The body's damping value is added to any value set in areas or the default value.

BodyDampMode **BODY_DAMP_MODE_REPLACE** = `1`

The body's damping value replaces any value set in areas or the default value.

enum **BodyState**: 🔗

BodyState **BODY_STATE_TRANSFORM** = `0`

Constant to set/get the current transform matrix of the body.

BodyState **BODY_STATE_LINEAR_VELOCITY** = `1`

Constant to set/get the current linear velocity of the body.

BodyState **BODY_STATE_ANGULAR_VELOCITY** = `2`

Constant to set/get the current angular velocity of the body.

BodyState **BODY_STATE_SLEEPING** = `3`

Constant to sleep/wake up a body, or to get whether it is sleeping.

BodyState **BODY_STATE_CAN_SLEEP** = `4`

Constant to set/get whether the body can sleep.

enum **AreaBodyStatus**: 🔗

AreaBodyStatus **AREA_BODY_ADDED** = `0`

The value of the first parameter and area callback function receives, when an object enters one of its shapes.

AreaBodyStatus **AREA_BODY_REMOVED** = 1

The value of the first parameter and area callback function receives, when an object exits one of its shapes.

enum **ProcessInfo**:

ProcessInfo **INFO_ACTIVE_OBJECTS** = 0

Constant to get the number of objects that are not sleeping.

ProcessInfo **INFO_COLLISION_PAIRS** = 1

Constant to get the number of possible collisions.

ProcessInfo **INFO_ISLAND_COUNT** = 2

Constant to get the number of space regions where a collision could occur.

enum **SpaceParameter**:

SpaceParameter **SPACE_PARAM_CONTACT_RECYCLE_RADIUS** = 0

Constant to set/get the maximum distance a pair of bodies has to move before their collision status has to be recalculated.

SpaceParameter **SPACE_PARAM_CONTACT_MAX_SEPARATION** = 1

Constant to set/get the maximum distance a shape can be from another before they are considered separated and the contact is discarded.

SpaceParameter **SPACE_PARAM_CONTACT_MAX_ALLOWED_PENETRATION** = 2

Constant to set/get the maximum distance a shape can penetrate another shape before it is considered a collision.

SpaceParameter **SPACE_PARAM_CONTACT_DEFAULT_BIAS** = 3

Constant to set/get the default solver bias for all physics contacts. A solver bias is a factor controlling how much two objects "rebound", after overlapping, to avoid leaving them in that state because of numerical imprecision.

SpaceParameter **SPACE_PARAM_BODY_LINEAR_VELOCITY_SLEEP_THRESHOLD** = `4`

Constant to set/get the threshold linear velocity of activity. A body marked as potentially inactive for both linear and angular velocity will be put to sleep after the time given.

SpaceParameter **SPACE_PARAM_BODY_ANGULAR_VELOCITY_SLEEP_THRESHOLD** = `5`

Constant to set/get the threshold angular velocity of activity. A body marked as potentially inactive for both linear and angular velocity will be put to sleep after the time given.

SpaceParameter **SPACE_PARAM_BODY_TIME_TO_SLEEP** = `6`

Constant to set/get the maximum time of activity. A body marked as potentially inactive for both linear and angular velocity will be put to sleep after this time.

SpaceParameter **SPACE_PARAM_SOLVER_ITERATIONS** = `7`

Constant to set/get the number of solver iterations for contacts and constraints. The greater the number of iterations, the more accurate the collisions and constraints will be. However, a greater number of iterations requires more CPU power, which can decrease performance.

---

enum **BodyAxis**: 🔗

BodyAxis **BODY_AXIS_LINEAR_X** = `1`

There is currently no description for this enum. Please help us by contributing one!

BodyAxis **BODY_AXIS_LINEAR_Y** = `2`

There is currently no description for this enum. Please help us by contributing one!

BodyAxis **BODY_AXIS_LINEAR_Z** = `4`

There is currently no description for this enum. Please help us by contributing one!

BodyAxis **BODY_AXIS_ANGULAR_X** = `8`

There is currently no description for this enum. Please help us by contributing one!

BodyAxis **BODY_AXIS_ANGULAR_Y** = `16`

There is currently no description for this enum. Please help us by contributing one!

BodyAxis **BODY_AXIS_ANGULAR_Z** = `32`

There is currently no description for this enum. Please help us by contributing one!

Method Descriptions

void **area_add_shape**(area: RID, shape: RID, transform: Transform3D = Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0), disabled: bool = false) 🔗

Adds a shape to the area, along with a transform matrix. Shapes are usually referenced by their index, so you should track which shape has a given index.

---

void **area_attach_object_instance_id**(area: RID, id: int) 🔗

Assigns the area to a descendant of Object, so it can exist in the node tree.

---

void **area_clear_shapes**(area: RID) 🔗

Removes all shapes from an area. It does not delete the shapes, so they can be reassigned later.

RID **area_create**() 🔗

Creates a 3D area object in the physics server, and returns the RID that identifies it. The default settings for the created area include a collision layer and mask set to `1`, and `monitorable` set to `false`.

Use area_add_shape to add shapes to it, use area_set_transform to set its transform, and use area_set_space to add the area to a space. If you want the area to be detectable use area_set_monitorable.

---

int **area_get_collision_layer**(area: RID) const 🔗

Returns the physics layer or layers an area belongs to.

---

int **area_get_collision_mask**(area: RID) const

Returns the physics layer or layers an area can contact with.

int **area_get_object_instance_id**(area: RID) const

Gets the instance ID of the object the area is assigned to.

Variant **area_get_param**(area: RID, param: AreaParameter) const

Returns an area parameter value. A list of available parameters is on the AreaParameter constants.

RID **area_get_shape**(area: RID, shape_idx: int) const

Returns the RID of the nth shape of an area.

int **area_get_shape_count**(area: RID) const

Returns the number of shapes assigned to an area.

Transform3D **area_get_shape_transform**(area: RID, shape_idx: int) const

Returns the transform matrix of a shape within an area.

RID **area_get_space**(area: RID) const

Returns the space assigned to the area.

Transform3D **area_get_transform**(area: RID) const

Returns the transform matrix for an area.

void **area_remove_shape**(area: RID, shape_idx: int)

Removes a shape from an area. It does not delete the shape, so it can be reassigned later.

---

void **area_set_area_monitor_callback**(area: RID, callback: Callable)

Sets the area's area monitor callback. This callback will be called when any other (shape of an) area enters or exits (a shape of) the given area, and must take the following five parameters:

1. an integer `status`: either AREA_BODY_ADDED or AREA_BODY_REMOVED depending on whether the other area's shape entered or exited the area,
2. an RID `area_rid`: the RID of the other area that entered or exited the area,
3. an integer `instance_id`: the `ObjectID` attached to the other area,
4. an integer `area_shape_idx`: the index of the shape of the other area that entered or exited the area,
5. an integer `self_shape_idx`: the index of the shape of the area where the other area entered or exited.

By counting (or keeping track of) the shapes that enter and exit, it can be determined if an area (with all its shapes) is entering for the first time or exiting for the last time.

---

void **area_set_collision_layer**(area: RID, layer: int)

Assigns the area to one or many physics layers.

---

void **area_set_collision_mask**(area: RID, mask: int)

Sets which physics layers the area will monitor.

---

void **area_set_monitor_callback**(area: RID, callback: Callable)

Sets the area's body monitor callback. This callback will be called when any other (shape of a) body enters or exits (a shape of) the given area, and must take the following five parameters:

1. an integer `status`: either AREA_BODY_ADDED or AREA_BODY_REMOVED depending on whether the other body shape entered or exited the area,
2. an RID `body_rid`: the RID of the body that entered or exited the area,
3. an integer `instance_id`: the `ObjectID` attached to the body,
4. an integer `body_shape_idx`: the index of the shape of the body that entered or exited the area,

5. an integer `self_shape_idx`: the index of the shape of the area where the body entered or exited.

By counting (or keeping track of) the shapes that enter and exit, it can be determined if a body (with all its shapes) is entering for the first time or exiting for the last time.

---

void **area_set_monitorable**(area: RID, monitorable: bool) 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **area_set_param**(area: RID, param: AreaParameter, value: Variant) 🔗

Sets the value for an area parameter. A list of available parameters is on the AreaParameter constants.

---

void **area_set_ray_pickable**(area: RID, enable: bool) 🔗

Sets object pickable with rays.

---

void **area_set_shape**(area: RID, shape_idx: int, shape: RID) 🔗

Substitutes a given area shape by another. The old shape is selected by its index, the new one by its RID.

---

void **area_set_shape_disabled**(area: RID, shape_idx: int, disabled: bool) 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **area_set_shape_transform**(area: RID, shape_idx: int, transform: Transform3D) 🔗

Sets the transform matrix for an area shape.

---

void **area_set_space**(area: RID, space: RID) 🔗

Assigns a space to the area.

---

void **area_set_transform**(area: RID, transform: Transform3D)

Sets the transform matrix for an area.

---

void **body_add_collision_exception**(body: RID, excepted_body: RID)

Adds a body to the list of bodies exempt from collisions.

---

void **body_add_constant_central_force**(body: RID, force: Vector3)

Adds a constant directional force without affecting rotation that keeps being applied over time until cleared with `body_set_constant_force(body, Vector3(0, 0, 0))`.

This is equivalent to using body_add_constant_force at the body's center of mass.

---

void **body_add_constant_force**(body: RID, force: Vector3, position: Vector3 = Vector3(0, 0, 0))

Adds a constant positioned force to the body that keeps being applied over time until cleared with `body_set_constant_force(body, Vector3(0, 0, 0))`.

`position` is the offset from the body origin in global coordinates.

---

void **body_add_constant_torque**(body: RID, torque: Vector3)

Adds a constant rotational force without affecting position that keeps being applied over time until cleared with `body_set_constant_torque(body, Vector3(0, 0, 0))`.

---

void **body_add_shape**(body: RID, shape: RID, transform: Transform3D = Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0), disabled: bool = false)

Adds a shape to the body, along with a transform matrix. Shapes are usually referenced by their index, so you should track which shape has a given index.

---

void **body_apply_central_force**(body: RID, force: Vector3)

Applies a directional force without affecting rotation. A force is time dependent and meant to be applied every physics update.

This is equivalent to using body_apply_force at the body's center of mass.

---

void **body_apply_central_impulse**(body: RID, impulse: Vector3)

Applies a directional impulse without affecting rotation.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

This is equivalent to using body_apply_impulse at the body's center of mass.

---

void **body_apply_force**(body: RID, force: Vector3, position: Vector3 = Vector3(0, 0, 0))

Applies a positioned force to the body. A force is time dependent and meant to be applied every physics update.

`position` is the offset from the body origin in global coordinates.

---

void **body_apply_impulse**(body: RID, impulse: Vector3, position: Vector3 = Vector3(0, 0, 0))

Applies a positioned impulse to the body.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

`position` is the offset from the body origin in global coordinates.

---

void **body_apply_torque**(body: RID, torque: Vector3)

Applies a rotational force without affecting position. A force is time dependent and meant to be applied every physics update.

---

void **body_apply_torque_impulse**(body: RID, impulse: Vector3)

Applies a rotational impulse to the body without affecting the position.

An impulse is time-independent! Applying an impulse every frame would result in a framerate-dependent force. For this reason, it should only be used when simulating one-time impacts (use the "_force" functions otherwise).

---

void **body_attach_object_instance_id**(body: RID, id: int)

Assigns the area to a descendant of Object, so it can exist in the node tree.

---

void **body_clear_shapes**(body: RID)

Removes all shapes from a body.

RID **body_create**()

Creates a 3D body object in the physics server, and returns the RID that identifies it. The default settings for the created area include a collision layer and mask set to `1`, and body mode set to BODY_MODE_RIGID.

Use body_add_shape to add shapes to it, use body_set_state to set its transform, and use body_set_space to add the body to a space.

---

int **body_get_collision_layer**(body: RID) const

Returns the physics layer or layers a body belongs to.

---

int **body_get_collision_mask**(body: RID) const

Returns the physics layer or layers a body can collide with.

---

float **body_get_collision_priority**(body: RID) const

Returns the body's collision priority.

Vector3 **body_get_constant_force**(body: RID) const

Returns the body's total constant positional forces applied during each physics update.

See body_add_constant_force and body_add_constant_central_force.

Vector3 **body_get_constant_torque**(body: RID) const

Returns the body's total constant rotational forces applied during each physics update.

See body_add_constant_torque.

PhysicsDirectBodyState3D **body_get_direct_state**(body: RID)

Returns the PhysicsDirectBodyState3D of the body. Returns `null` if the body is destroyed or removed from the physics space.

---

int **body_get_max_contacts_reported**(body: RID) const

Returns the maximum contacts that can be reported. See body_set_max_contacts_reported.

BodyMode **body_get_mode**(body: RID) const

Returns the body mode.

---

int **body_get_object_instance_id**(body: RID) const

Gets the instance ID of the object the area is assigned to.

Variant **body_get_param**(body: RID, param: BodyParameter) const

Returns the value of a body parameter. A list of available parameters is on the BodyParameter constants.

RID **body_get_shape**(body: RID, shape_idx: int) const

Returns the RID of the nth shape of a body.

---

int **body_get_shape_count**(body: RID) const

Returns the number of shapes assigned to a body.

Transform3D **body_get_shape_transform**(body: RID, shape_idx: int) const

Returns the transform matrix of a body shape.

RID **body_get_space**(body: RID) const

Returns the RID of the space assigned to a body.

Variant **body_get_state**(body: RID, state: BodyState) const

Returns a body state.

---

bool **body_is_axis_locked**(body: RID, axis: BodyAxis) const

There is currently no description for this method. Please help us by contributing one!

---

bool **body_is_continuous_collision_detection_enabled**(body: RID) const

If `true`, the continuous collision detection mode is enabled.

---

bool **body_is_omitting_force_integration**(body: RID) const

Returns `true` if the body is omitting the standard force integration. See body_set_omit_force_integration.

---

void **body_remove_collision_exception**(body: RID, excepted_body: RID)

Removes a body from the list of bodies exempt from collisions.

Continuous collision detection tries to predict where a moving body will collide, instead of moving it and correcting its movement if it collided.

---

void **body_remove_shape**(body: RID, shape_idx: int)

Removes a shape from a body. The shape is not deleted, so it can be reused afterwards.

---

void **body_reset_mass_properties**(body: RID)

Restores the default inertia and center of mass based on shapes to cancel any custom values previously set using body_set_param.

---

void **body_set_axis_lock**(body: RID, axis: BodyAxis, lock: bool)

There is currently no description for this method. Please help us by contributing one!

---

void **body_set_axis_velocity**(body: RID, axis_velocity: Vector3)

Sets an axis velocity. The velocity in the given vector axis will be set as the given vector length. This is useful for jumping behavior.

---

void **body_set_collision_layer**(body: RID, layer: int)

Sets the physics layer or layers a body belongs to.

---

void **body_set_collision_mask**(body: RID, mask: int)

Sets the physics layer or layers a body can collide with.

---

void **body_set_collision_priority**(body: RID, priority: float)

Sets the body's collision priority.

---

void **body_set_constant_force**(body: RID, force: Vector3)

Sets the body's total constant positional forces applied during each physics update.

See body_add_constant_force and body_add_constant_central_force.

---

void **body_set_constant_torque**(body: RID, torque: Vector3) 🔗

Sets the body's total constant rotational forces applied during each physics update.

See body_add_constant_torque.

---

void **body_set_enable_continuous_collision_detection**(body: RID, enable: bool) 🔗

If `true`, the continuous collision detection mode is enabled.

Continuous collision detection tries to predict where a moving body will collide, instead of moving it and correcting its movement if it collided.

---

void **body_set_force_integration_callback**(body: RID, callable: Callable, userdata: Variant = null) 🔗

Sets the body's custom force integration callback function to `callable`. Use an empty Callable (`Callable()`) to clear the custom callback.

The function `callable` will be called every physics tick, before the standard force integration (see body_set_omit_force_integration). It can be used for example to update the body's linear and angular velocity based on contact with other bodies.

If `userdata` is not `null`, the function `callable` must take the following two parameters:

1. `state`: a PhysicsDirectBodyState3D, used to retrieve and modify the body's state,
2. `userdata`: a Variant; its value will be the `userdata` passed into this method.

If `userdata` is `null`, then `callable` must take only the `state` parameter.

---

void **body_set_max_contacts_reported**(body: RID, amount: int) 🔗

Sets the maximum contacts to report. Bodies can keep a log of the contacts with other bodies. This is enabled by setting the maximum number of contacts reported to a number greater than 0.

---

void **body_set_mode**(body: RID, mode: BodyMode) 🔗

Sets the body mode, from one of the BodyMode constants.

---

void **body_set_omit_force_integration**(body: RID, enable: bool)

Sets whether the body omits the standard force integration. If `enable` is `true`, the body will not automatically use applied forces, torques, and damping to update the body's linear and angular velocity. In this case, body_set_force_integration_callback can be used to manually update the linear and angular velocity instead.

This method is called when the property RigidBody3D.custom_integrator is set.

---

void **body_set_param**(body: RID, param: BodyParameter, value: Variant)

Sets a body parameter. A list of available parameters is on the BodyParameter constants.

---

void **body_set_ray_pickable**(body: RID, enable: bool)

Sets the body pickable with rays if `enable` is set.

---

void **body_set_shape**(body: RID, shape_idx: int, shape: RID)

Substitutes a given body shape by another. The old shape is selected by its index, the new one by its RID.

---

void **body_set_shape_disabled**(body: RID, shape_idx: int, disabled: bool)

There is currently no description for this method. Please help us by contributing one!

---

void **body_set_shape_transform**(body: RID, shape_idx: int, transform: Transform3D)

Sets the transform matrix for a body shape.

---

void **body_set_space**(body: RID, space: RID)

Assigns a space to the body (see space_create).

---

void **body_set_state**(body: RID, state: BodyState, value: Variant) 🔗

Sets a body state (see BodyState constants).

---

void **body_set_state_sync_callback**(body: RID, callable: Callable) 🔗

Sets the body's state synchronization callback function to `callable`. Use an empty Callable (`Callable()`) to clear the callback.

The function `callable` will be called every physics frame, assuming that the body was active during the previous physics tick, and can be used to fetch the latest state from the physics server.

The function `callable` must take the following parameters:

1. `state`: a PhysicsDirectBodyState3D, used to retrieve the body's state.

---

bool **body_test_motion**(body: RID, parameters: PhysicsTestMotionParameters3D, result: PhysicsTestMotionResult3D = null) 🔗

Returns `true` if a collision would result from moving along a motion vector from a given point in space. PhysicsTestMotionParameters3D is passed to set motion parameters. PhysicsTestMotionResult3D can be passed to return additional information.

RID **box_shape_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

RID **capsule_shape_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

RID **concave_polygon_shape_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **cone_twist_joint_get_param**(joint: RID, param: ConeTwistJointParam) const 🔗

Gets a cone_twist_joint parameter (see ConeTwistJointParam constants).

---

void **cone_twist_joint_set_param**(joint: RID, param: ConeTwistJointParam, value: float) 🔗

Sets a cone_twist_joint parameter (see ConeTwistJointParam constants).

RID **convex_polygon_shape_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

RID **custom_shape_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

RID **cylinder_shape_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **free_rid**(rid: RID) 🔗

Destroys any of the objects created by PhysicsServer3D. If the RID passed is not one of the objects that can be created by PhysicsServer3D, an error will be sent to the console.

---

bool **generic_6dof_joint_get_flag**(joint: RID, axis: Vector3.Axis, flag: G6DOFJointAxisFlag) const 🔗

Returns the value of a generic 6DOF joint flag. See G6DOFJointAxisFlag for the list of available flags.

---

float **generic_6dof_joint_get_param**(joint: RID, axis: Vector3.Axis, param: G6DOFJointAxisParam) const

Returns the value of a generic 6DOF joint parameter. See G6DOFJointAxisParam for the list of available parameters.

void **generic_6dof_joint_set_flag**(joint: RID, axis: Vector3.Axis, flag: G6DOFJointAxisFlag, enable: bool)

Sets the value of a given generic 6DOF joint flag. See G6DOFJointAxisFlag for the list of available flags.

void **generic_6dof_joint_set_param**(joint: RID, axis: Vector3.Axis, param: G6DOFJointAxisParam, value: float)

Sets the value of a given generic 6DOF joint parameter. See G6DOFJointAxisParam for the list of available parameters.

int **get_process_info**(process_info: ProcessInfo)

Returns information about the current state of the 3D physics engine. See ProcessInfo for a list of available states.

RID **heightmap_shape_create**()

There is currently no description for this method. Please help us by contributing one!

bool **hinge_joint_get_flag**(joint: RID, flag: HingeJointFlag) const

Gets a hinge_joint flag (see HingeJointFlag constants).

float **hinge_joint_get_param**(joint: RID, param: HingeJointParam) const

Gets a hinge_joint parameter (see HingeJointParam).

void **hinge_joint_set_flag**(joint: RID, flag: HingeJointFlag, enabled: bool)

Sets a hinge_joint flag (see HingeJointFlag constants).

void **hinge_joint_set_param**(joint: RID, param: HingeJointParam, value: float) 🔗

Sets a hinge_joint parameter (see HingeJointParam constants).

void **joint_clear**(joint: RID) 🔗

There is currently no description for this method. Please help us by contributing one!

RID **joint_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

void **joint_disable_collisions_between_bodies**(joint: RID, disable: bool) 🔗

Sets whether the bodies attached to the Joint3D will collide with each other.

int **joint_get_solver_priority**(joint: RID) const 🔗

Gets the priority value of the Joint3D.

JointType **joint_get_type**(joint: RID) const 🔗

Returns the type of the Joint3D.

bool **joint_is_disabled_collisions_between_bodies**(joint: RID) const 🔗

Returns whether the bodies attached to the Joint3D will collide with each other.

void **joint_make_cone_twist**(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D) 🔗

There is currently no description for this method. Please help us by contributing one!

void **joint_make_generic_6dof**(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D)

Make the joint a generic six degrees of freedom (6DOF) joint. Use generic_6dof_joint_set_flag and generic_6dof_joint_set_param to set the joint's flags and parameters respectively.

void **joint_make_hinge**(joint: RID, body_A: RID, hinge_A: Transform3D, body_B: RID, hinge_B: Transform3D)

There is currently no description for this method. Please help us by contributing one!

void **joint_make_pin**(joint: RID, body_A: RID, local_A: Vector3, body_B: RID, local_B: Vector3)

There is currently no description for this method. Please help us by contributing one!

void **joint_make_slider**(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D)

There is currently no description for this method. Please help us by contributing one!

void **joint_set_solver_priority**(joint: RID, priority: int)

Sets the priority value of the Joint3D.

Vector3 **pin_joint_get_local_a**(joint: RID) const

Returns position of the joint in the local space of body a of the joint.

Vector3 **pin_joint_get_local_b**(joint: RID) const

Returns position of the joint in the local space of body b of the joint.

---

float **pin_joint_get_param**(joint: RID, param: PinJointParam) const 🔗

Gets a pin_joint parameter (see PinJointParam constants).

---

void **pin_joint_set_local_a**(joint: RID, local_A: Vector3) 🔗

Sets position of the joint in the local space of body a of the joint.

---

void **pin_joint_set_local_b**(joint: RID, local_B: Vector3) 🔗

Sets position of the joint in the local space of body b of the joint.

---

void **pin_joint_set_param**(joint: RID, param: PinJointParam, value: float) 🔗

Sets a pin_joint parameter (see PinJointParam constants).

RID **separation_ray_shape_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **set_active**(active: bool) 🔗

Activates or deactivates the 3D physics engine.

Variant **shape_get_data**(shape: RID) const 🔗

Returns the shape data.

---

float **shape_get_margin**(shape: RID) const 🔗

Returns the collision margin for the shape.

Note: This is not used in Godot Physics, so will always return `0`.

ShapeType **shape_get_type**(shape: RID) const

Returns the type of shape (see ShapeType constants).

---

void **shape_set_data**(shape: RID, data: Variant)

Sets the shape data that defines its shape and size. The data to be passed depends on the kind of shape created shape_get_type.

---

void **shape_set_margin**(shape: RID, margin: float)

Sets the collision margin for the shape.

Note: This is not used in Godot Physics.

---

float **slider_joint_get_param**(joint: RID, param: SliderJointParam) const

Gets a slider_joint parameter (see SliderJointParam constants).

---

void **slider_joint_set_param**(joint: RID, param: SliderJointParam, value: float)

Gets a slider_joint parameter (see SliderJointParam constants).

---

void **soft_body_add_collision_exception**(body: RID, body_b: RID)

Adds the given body to the list of bodies exempt from collisions.

RID **soft_body_create**()

Creates a new soft body and returns its internal RID.

AABB **soft_body_get_bounds**(body: RID) const

Returns the bounds of the given soft body in global coordinates.

---

int **soft_body_get_collision_layer**(body: RID) const

Returns the physics layer or layers that the given soft body belongs to.

---

int **soft_body_get_collision_mask**(body: RID) const

Returns the physics layer or layers that the given soft body can collide with.

---

float **soft_body_get_damping_coefficient**(body: RID) const

Returns the damping coefficient of the given soft body.

---

float **soft_body_get_drag_coefficient**(body: RID) const

Returns the drag coefficient of the given soft body.

---

float **soft_body_get_linear_stiffness**(body: RID) const

Returns the linear stiffness of the given soft body.

Vector3 **soft_body_get_point_global_position**(body: RID, point_index: int) const

Returns the current position of the given soft body point in global coordinates.

---

float **soft_body_get_pressure_coefficient**(body: RID) const

Returns the pressure coefficient of the given soft body.

---

int **soft_body_get_simulation_precision**(body: RID) const

Returns the simulation precision of the given soft body.

RID **soft_body_get_space**(body: RID) const

Returns the RID of the space assigned to the given soft body.

Variant **soft_body_get_state**(body: RID, state: BodyState) const

Returns the given soft body state (see BodyState constants).

Note: Godot's default physics implementation does not support BODY_STATE_LINEAR_VELOCITY, BODY_STATE_ANGULAR_VELOCITY, BODY_STATE_SLEEPING, or BODY_STATE_CAN_SLEEP.

float **soft_body_get_total_mass**(body: RID) const

Returns the total mass assigned to the given soft body.

bool **soft_body_is_point_pinned**(body: RID, point_index: int) const

Returns whether the given soft body point is pinned.

void **soft_body_move_point**(body: RID, point_index: int, global_position: Vector3)

Moves the given soft body point to a position in global coordinates.

void **soft_body_pin_point**(body: RID, point_index: int, pin: bool)

Pins or unpins the given soft body point based on the value of `pin`.

Note: Pinning a point effectively makes it kinematic, preventing it from being affected by forces, but you can still move it using soft_body_move_point.

void **soft_body_remove_all_pinned_points**(body: RID)

Unpins all points of the given soft body.

void **soft_body_remove_collision_exception**(body: RID, body_b: RID)

Removes the given body from the list of bodies exempt from collisions.

void **soft_body_set_collision_layer**(body: RID, layer: int)

Sets the physics layer or layers the given soft body belongs to.

void **soft_body_set_collision_mask**(body: RID, mask: int)

Sets the physics layer or layers the given soft body can collide with.

void **soft_body_set_damping_coefficient**(body: RID, damping_coefficient: float)

Sets the damping coefficient of the given soft body. Higher values will slow down the body more noticeably when forces are applied.

void **soft_body_set_drag_coefficient**(body: RID, drag_coefficient: float)

Sets the drag coefficient of the given soft body. Higher values increase this body's air resistance.

Note: This value is currently unused by Godot's default physics implementation.

void **soft_body_set_linear_stiffness**(body: RID, stiffness: float)

Sets the linear stiffness of the given soft body. Higher values will result in a stiffer body, while lower values will increase the body's ability to bend. The value can be between `0.0` and `1.0` (inclusive).

void **soft_body_set_mesh**(body: RID, mesh: RID)

Sets the mesh of the given soft body.

void **soft_body_set_pressure_coefficient**(body: RID, pressure_coefficient: float)

Sets the pressure coefficient of the given soft body. Simulates pressure build-up from inside this body. Higher values increase the strength of this effect.

void **soft_body_set_ray_pickable**(body: RID, enable: bool)

Sets whether the given soft body will be pickable when using object picking.

void **soft_body_set_simulation_precision**(body: RID, simulation_precision: int)

Sets the simulation precision of the given soft body. Increasing this value will improve the resulting simulation, but can affect performance. Use with care.

---

void **soft_body_set_space**(body: RID, space: RID)

Assigns a space to the given soft body (see space_create).

---

void **soft_body_set_state**(body: RID, state: BodyState, variant: Variant)

Sets the given body state for the given body (see BodyState constants).

Note: Godot's default physics implementation does not support BODY_STATE_LINEAR_VELOCITY, BODY_STATE_ANGULAR_VELOCITY, BODY_STATE_SLEEPING, or BODY_STATE_CAN_SLEEP.

---

void **soft_body_set_total_mass**(body: RID, total_mass: float)

Sets the total mass for the given soft body.

---

void **soft_body_set_transform**(body: RID, transform: Transform3D)

Sets the global transform of the given soft body.

---

void **soft_body_update_rendering_server**(body: RID, rendering_server_handler: PhysicsServer3DRenderingServerHandler)

Requests that the physics server updates the rendering server with the latest positions of the given soft body's points through the `rendering_server_handler` interface.

RID **space_create**()

Creates a space. A space is a collection of parameters for the physics engine that can be assigned to an area or a body. It can be assigned to an area with area_set_space, or to a body with body_set_space.

PhysicsDirectSpaceState3D **space_get_direct_state**(space: RID) 🔗

Returns the state of a space, a PhysicsDirectSpaceState3D. This object can be used to make collision/intersection queries.

---

float **space_get_param**(space: RID, param: SpaceParameter) const 🔗

Returns the value of a space parameter.

---

bool **space_is_active**(space: RID) const 🔗

Returns whether the space is active.

---

void **space_set_active**(space: RID, active: bool) 🔗

Marks a space as active. It will not have an effect, unless it is assigned to an area or body.

---

void **space_set_param**(space: RID, param: SpaceParameter, value: float) 🔗

Sets the value for a space parameter. A list of available parameters is on the SpaceParameter constants.

RID **sphere_shape_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

RID **world_boundary_shape_create**() 🔗

There is currently no description for this method. Please help us by contributing one!

PhysicsServer3DExtension

Inherits: PhysicsServer3D < Object

Provides virtual methods that can be overridden to create custom PhysicsServer3D implementations.

Description

This class extends PhysicsServer3D by providing additional virtual methods that can be overridden. When these methods are overridden, they will be called instead of the internal methods of the physics server.

Intended for use with GDExtension to create custom implementations of PhysicsServer3D.

Methods

void	_area_add_shape(area: RID, shape: RID, transform: Transform3D, disabled: bool) virtual
void	_area_attach_object_instance_id(area: RID, id: int) virtual
void	_area_clear_shapes(area: RID) virtual
RID	_area_create() virtual
int	_area_get_collision_layer(area: RID) virtual const
int	_area_get_collision_mask(area: RID) virtual const
int	_area_get_object_instance_id(area: RID) virtual const

Variant	_area_get_param(area: RID, param: AreaParameter) virtual const
RID	_area_get_shape(area: RID, shape_idx: int) virtual const
int	_area_get_shape_count(area: RID) virtual const
Transform3D	_area_get_shape_transform(area: RID, shape_idx: int) virtual const
RID	_area_get_space(area: RID) virtual const
Transform3D	_area_get_transform(area: RID) virtual const
void	_area_remove_shape(area: RID, shape_idx: int) virtual
void	_area_set_area_monitor_callback(area: RID, callback: Callable) virtual
void	_area_set_collision_layer(area: RID, layer: int) virtual
void	_area_set_collision_mask(area: RID, mask: int) virtual
void	_area_set_monitor_callback(area: RID, callback: Callable) virtual
void	_area_set_monitorable(area: RID, monitorable: bool) virtual
void	_area_set_param(area: RID, param: AreaParameter, value: Variant) virtual

void	_area_set_ray_pickable(area: RID, enable: bool) virtual
void	_area_set_shape(area: RID, shape_idx: int, shape: RID) virtual
void	_area_set_shape_disabled(area: RID, shape_idx: int, disabled: bool) virtual
void	_area_set_shape_transform(area: RID, shape_idx: int, transform: Transform3D) virtual
void	_area_set_space(area: RID, space: RID) virtual
void	_area_set_transform(area: RID, transform: Transform3D) virtual
void	_body_add_collision_exception(body: RID, excepted_body: RID) virtual
void	_body_add_constant_central_force(body: RID, force: Vector3) virtual
void	_body_add_constant_force(body: RID, force: Vector3, position: Vector3) virtual
void	_body_add_constant_torque(body: RID, torque: Vector3) virtual
void	_body_add_shape(body: RID, shape: RID, transform: Transform3D, disabled: bool) virtual
void	_body_apply_central_force(body: RID, force: Vector3) virtual

void	_body_apply_central_impulse(body: RID, impulse: Vector3) virtual
void	_body_apply_force(body: RID, force: Vector3, position: Vector3) virtual
void	_body_apply_impulse(body: RID, impulse: Vector3, position: Vector3) virtual
void	_body_apply_torque(body: RID, torque: Vector3) virtual
void	_body_apply_torque_impulse(body: RID, impulse: Vector3) virtual
void	_body_attach_object_instance_id(body: RID, id: int) virtual
void	_body_clear_shapes(body: RID) virtual
RID	_body_create() virtual
Array[RID]	_body_get_collision_exceptions(body: RID) virtual const
int	_body_get_collision_layer(body: RID) virtual const
int	_body_get_collision_mask(body: RID) virtual const
float	_body_get_collision_priority(body: RID) virtual const
Vector3	_body_get_constant_force(body: RID) virtual const
Vector3	_body_get_constant_torque(body: RID) virtual const

float	_body_get_contacts_reported_depth_threshold(body: RID) virtual const
PhysicsDirectBodyState3D	_body_get_direct_state(body: RID) virtual
int	_body_get_max_contacts_reported(body: RID) virtual const
BodyMode	_body_get_mode(body: RID) virtual const
int	_body_get_object_instance_id(body: RID) virtual const
Variant	_body_get_param(body: RID, param: BodyParameter) virtual const
RID	_body_get_shape(body: RID, shape_idx: int) virtual const
int	_body_get_shape_count(body: RID) virtual const
Transform3D	_body_get_shape_transform(body: RID, shape_idx: int) virtual const
RID	_body_get_space(body: RID) virtual const
Variant	_body_get_state(body: RID, state: BodyState) virtual const
int	_body_get_user_flags(body: RID) virtual const
bool	_body_is_axis_locked(body: RID, axis: BodyAxis) virtual const

bool	_body_is_continuous_collision_detection_enabled(body: RID) virtual const
bool	_body_is_omitting_force_integration(body: RID) virtual const
void	_body_remove_collision_exception(body: RID, excepted_body: RID) virtual
void	_body_remove_shape(body: RID, shape_idx: int) virtual
void	_body_reset_mass_properties(body: RID) virtual
void	_body_set_axis_lock(body: RID, axis: BodyAxis, lock: bool) virtual
void	_body_set_axis_velocity(body: RID, axis_velocity: Vector3) virtual
void	_body_set_collision_layer(body: RID, layer: int) virtual
void	_body_set_collision_mask(body: RID, mask: int) virtual
void	_body_set_collision_priority(body: RID, priority: float) virtual
void	_body_set_constant_force(body: RID, force: Vector3) virtual
void	_body_set_constant_torque(body: RID, torque: Vector3) virtual

void	_body_set_contacts_reported_depth_threshold(body: RID, threshold: float) virtual
void	_body_set_enable_continuous_collision_detection(body: RID, enable: bool) virtual
void	_body_set_force_integration_callback(body: RID, callable: Callable, userdata: Variant) virtual
void	_body_set_max_contacts_reported(body: RID, amount: int) virtual
void	_body_set_mode(body: RID, mode: BodyMode) virtual
void	_body_set_omit_force_integration(body: RID, enable: bool) virtual
void	_body_set_param(body: RID, param: BodyParameter, value: Variant) virtual
void	_body_set_ray_pickable(body: RID, enable: bool) virtual
void	_body_set_shape(body: RID, shape_idx: int, shape: RID) virtual
void	_body_set_shape_disabled(body: RID, shape_idx: int, disabled: bool) virtual
void	_body_set_shape_transform(body: RID, shape_idx: int, transform: Transform3D) virtual
void	_body_set_space(body: RID, space: RID) virtual

void	_body_set_state(body: RID, state: BodyState, value: Variant) virtual
void	_body_set_state_sync_callback(body: RID, callable: Callable) virtual
void	_body_set_user_flags(body: RID, flags: int) virtual
bool	_body_test_motion(body: RID, from: Transform3D, motion: Vector3, margin: float, max_collisions: int, collide_separation_ray: bool, recovery_as_collision: bool, result: `PhysicsServer3DExtensionMotionResult*`) virtual const
RID	_box_shape_create() virtual
RID	_capsule_shape_create() virtual
RID	_concave_polygon_shape_create() virtual
float	_cone_twist_joint_get_param(joint: RID, param: ConeTwistJointParam) virtual const
void	_cone_twist_joint_set_param(joint: RID, param: ConeTwistJointParam, value: float) virtual
RID	_convex_polygon_shape_create() virtual
RID	_custom_shape_create() virtual
RID	_cylinder_shape_create() virtual

void	_end_sync() virtual
void	_finish() virtual
void	_flush_queries() virtual
void	_free_rid(rid: RID) virtual
bool	_generic_6dof_joint_get_flag(joint: RID, axis: Vector3.Axis, flag: G6DOFJointAxisFlag) virtual const
float	_generic_6dof_joint_get_param(joint: RID, axis: Vector3.Axis, param: G6DOFJointAxisParam) virtual const
void	_generic_6dof_joint_set_flag(joint: RID, axis: Vector3.Axis, flag: G6DOFJointAxisFlag, enable: bool) virtual
void	_generic_6dof_joint_set_param(joint: RID, axis: Vector3.Axis, param: G6DOFJointAxisParam, value: float) virtual
int	_get_process_info(process_info: ProcessInfo) virtual
RID	_heightmap_shape_create() virtual
bool	_hinge_joint_get_flag(joint: RID, flag: HingeJointFlag) virtual const
float	_hinge_joint_get_param(joint: RID, param: HingeJointParam) virtual const

void	_hinge_joint_set_flag(joint: RID, flag: HingeJointFlag, enabled: bool) virtual
void	_hinge_joint_set_param(joint: RID, param: HingeJointParam, value: float) virtual
void	_init() virtual
bool	_is_flushing_queries() virtual const
void	_joint_clear(joint: RID) virtual
RID	_joint_create() virtual
void	_joint_disable_collisions_between_bodies(joint: RID, disable: bool) virtual
int	_joint_get_solver_priority(joint: RID) virtual const
JointType	_joint_get_type(joint: RID) virtual const
bool	_joint_is_disabled_collisions_between_bodies(joint: RID) virtual const
void	_joint_make_cone_twist(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D) virtual
void	_joint_make_generic_6dof(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D) virtual

void	_joint_make_hinge(joint: RID, body_A: RID, hinge_A: Transform3D, body_B: RID, hinge_B: Transform3D) virtual
void	_joint_make_hinge_simple(joint: RID, body_A: RID, pivot_A: Vector3, axis_A: Vector3, body_B: RID, pivot_B: Vector3, axis_B: Vector3) virtual
void	_joint_make_pin(joint: RID, body_A: RID, local_A: Vector3, body_B: RID, local_B: Vector3) virtual
void	_joint_make_slider(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D) virtual
void	_joint_set_solver_priority(joint: RID, priority: int) virtual
Vector3	_pin_joint_get_local_a(joint: RID) virtual const
Vector3	_pin_joint_get_local_b(joint: RID) virtual const
float	_pin_joint_get_param(joint: RID, param: PinJointParam) virtual const
void	_pin_joint_set_local_a(joint: RID, local_A: Vector3) virtual
void	_pin_joint_set_local_b(joint: RID, local_B: Vector3) virtual
void	_pin_joint_set_param(joint: RID, param: PinJointParam, value: float) virtual
RID	_separation_ray_shape_create() virtual

void	_set_active(active: bool) virtual
float	_shape_get_custom_solver_bias(shape: RID) virtual const
Variant	_shape_get_data(shape: RID) virtual const
float	_shape_get_margin(shape: RID) virtual const
ShapeType	_shape_get_type(shape: RID) virtual const
void	_shape_set_custom_solver_bias(shape: RID, bias: float) virtual
void	_shape_set_data(shape: RID, data: Variant) virtual
void	_shape_set_margin(shape: RID, margin: float) virtual
float	_slider_joint_get_param(joint: RID, param: SliderJointParam) virtual const
void	_slider_joint_set_param(joint: RID, param: SliderJointParam, value: float) virtual
void	_soft_body_add_collision_exception(body: RID, body_b: RID) virtual
RID	_soft_body_create() virtual
AABB	_soft_body_get_bounds(body: RID) virtual const

Array[RID]	_soft_body_get_collision_exceptions(body: RID) virtual const
int	_soft_body_get_collision_layer(body: RID) virtual const
int	_soft_body_get_collision_mask(body: RID) virtual const
float	_soft_body_get_damping_coefficient(body: RID) virtual const
float	_soft_body_get_drag_coefficient(body: RID) virtual const
float	_soft_body_get_linear_stiffness(body: RID) virtual const
Vector3	_soft_body_get_point_global_position(body: RID, point_index: int) virtual const
float	_soft_body_get_pressure_coefficient(body: RID) virtual const
int	_soft_body_get_simulation_precision(body: RID) virtual const
RID	_soft_body_get_space(body: RID) virtual const
Variant	_soft_body_get_state(body: RID, state: BodyState) virtual const
float	_soft_body_get_total_mass(body: RID) virtual const

bool	_soft_body_is_point_pinned(body: RID, point_index: int) virtual const
void	_soft_body_move_point(body: RID, point_index: int, global_position: Vector3) virtual
void	_soft_body_pin_point(body: RID, point_index: int, pin: bool) virtual
void	_soft_body_remove_all_pinned_points(body: RID) virtual
void	_soft_body_remove_collision_exception(body: RID, body_b: RID) virtual
void	_soft_body_set_collision_layer(body: RID, layer: int) virtual
void	_soft_body_set_collision_mask(body: RID, mask: int) virtual
void	_soft_body_set_damping_coefficient(body: RID, damping_coefficient: float) virtual
void	_soft_body_set_drag_coefficient(body: RID, drag_coefficient: float) virtual
void	_soft_body_set_linear_stiffness(body: RID, linear_stiffness: float) virtual
void	_soft_body_set_mesh(body: RID, mesh: RID) virtual
void	_soft_body_set_pressure_coefficient(body: RID, pressure_coefficient: float) virtual

void	_soft_body_set_ray_pickable(body: RID, enable: bool) virtual
void	_soft_body_set_simulation_precision(body: RID, simulation_precision: int) virtual
void	_soft_body_set_space(body: RID, space: RID) virtual
void	_soft_body_set_state(body: RID, state: BodyState, variant: Variant) virtual
void	_soft_body_set_total_mass(body: RID, total_mass: float) virtual
void	_soft_body_set_transform(body: RID, transform: Transform3D) virtual
void	_soft_body_update_rendering_server(body: RID, rendering_server_handler: PhysicsServer3DRenderingServerHandler) virtual
RID	_space_create() virtual
int	_space_get_contact_count(space: RID) virtual const
PackedVector3Array	_space_get_contacts(space: RID) virtual const
PhysicsDirectSpaceState3D	_space_get_direct_state(space: RID) virtual
float	_space_get_param(space: RID, param: SpaceParameter) virtual const
bool	_space_is_active(space: RID) virtual const

void	_space_set_active(space: RID, active: bool) virtual
void	_space_set_debug_contacts(space: RID, max_contacts: int) virtual
void	_space_set_param(space: RID, param: SpaceParameter, value: float) virtual
RID	_sphere_shape_create() virtual
void	_step(step: float) virtual
void	_sync() virtual
RID	_world_boundary_shape_create() virtual
bool	body_test_motion_is_excluding_body(body: RID) const
bool	body_test_motion_is_excluding_object(object: int) const

Method Descriptions

void **_area_add_shape**(area: RID, shape: RID, transform: Transform3D, disabled: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_area_attach_object_instance_id**(area: RID, id: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_area_clear_shapes**(area: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_area_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_area_get_collision_layer**(area: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_area_get_collision_mask**(area: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_area_get_object_instance_id**(area: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Variant **_area_get_param**(area: RID, param: AreaParameter) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_area_get_shape**(area: RID, shape_idx: int) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_area_get_shape_count**(area: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Transform3D **_area_get_shape_transform**(area: RID, shape_idx: int) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_area_get_space**(area: RID) virtual const

There is currently no description for this method. Please help us by contributing one!

Transform3D **_area_get_transform**(area: RID) virtual const

There is currently no description for this method. Please help us by contributing one!

---

void **_area_remove_shape**(area: RID, shape_idx: int) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_area_monitor_callback**(area: RID, callback: Callable) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_collision_layer**(area: RID, layer: int) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_collision_mask**(area: RID, mask: int) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_monitor_callback**(area: RID, callback: Callable) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_monitorable**(area: RID, monitorable: bool) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_param**(area: RID, param: AreaParameter, value: Variant) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_ray_pickable**(area: RID, enable: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_shape**(area: RID, shape_idx: int, shape: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_shape_disabled**(area: RID, shape_idx: int, disabled: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_shape_transform**(area: RID, shape_idx: int, transform: Transform3D) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_space**(area: RID, space: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_area_set_transform**(area: RID, transform: Transform3D) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_add_collision_exception**(body: RID, excepted_body: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_add_constant_central_force**(body: RID, force: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_add_constant_force**(body: RID, force: Vector3, position: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_body_add_constant_torque**(body: RID, torque: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_add_shape**(body: RID, shape: RID, transform: Transform3D, disabled: bool) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_apply_central_force**(body: RID, force: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_apply_central_impulse**(body: RID, impulse: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_apply_force**(body: RID, force: Vector3, position: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_apply_impulse**(body: RID, impulse: Vector3, position: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_apply_torque**(body: RID, torque: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_apply_torque_impulse**(body: RID, impulse: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_attach_object_instance_id**(body: RID, id: int) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_body_clear_shapes**(body: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_body_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

Array[RID] **_body_get_collision_exceptions**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_body_get_collision_layer**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_body_get_collision_mask**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_body_get_collision_priority**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_body_get_constant_force**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_body_get_constant_torque**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_body_get_contacts_reported_depth_threshold**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

PhysicsDirectBodyState3D **_body_get_direct_state**(body: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_body_get_max_contacts_reported**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

BodyMode **_body_get_mode**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_body_get_object_instance_id**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Variant **_body_get_param**(body: RID, param: BodyParameter) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_body_get_shape**(body: RID, shape_idx: int) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_body_get_shape_count**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Transform3D **_body_get_shape_transform**(body: RID, shape_idx: int) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_body_get_space**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Variant **_body_get_state**(body: RID, state: BodyState) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_body_get_user_flags**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_body_is_axis_locked**(body: RID, axis: BodyAxis) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_body_is_continuous_collision_detection_enabled**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_body_is_omitting_force_integration**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_remove_collision_exception**(body: RID, excepted_body: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_remove_shape**(body: RID, shape_idx: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_reset_mass_properties**(body: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_axis_lock**(body: RID, axis: BodyAxis, lock: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_axis_velocity**(body: RID, axis_velocity: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_collision_layer**(body: RID, layer: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_collision_mask**(body: RID, mask: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_collision_priority**(body: RID, priority: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_constant_force**(body: RID, force: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_constant_torque**(body: RID, torque: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_contacts_reported_depth_threshold**(body: RID, threshold: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_enable_continuous_collision_detection**(body: RID, enable: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_force_integration_callback**(body: RID, callable: Callable, userdata: Variant) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_body_set_max_contacts_reported**(body: RID, amount: int) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_set_mode**(body: RID, mode: BodyMode) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_set_omit_force_integration**(body: RID, enable: bool) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_set_param**(body: RID, param: BodyParameter, value: Variant) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_set_ray_pickable**(body: RID, enable: bool) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_set_shape**(body: RID, shape_idx: int, shape: RID) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_set_shape_disabled**(body: RID, shape_idx: int, disabled: bool) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_set_shape_transform**(body: RID, shape_idx: int, transform: Transform3D) virtual

There is currently no description for this method. Please help us by contributing one!

void **_body_set_space**(body: RID, space: RID) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_state**(body: RID, state: BodyState, value: Variant) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_state_sync_callback**(body: RID, callable: Callable) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_body_set_user_flags**(body: RID, flags: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_body_test_motion**(body: RID, from: Transform3D, motion: Vector3, margin: float, max_collisions: int, collide_separation_ray: bool, recovery_as_collision: bool, result: `PhysicsServer3DExtensionMotionResult*`) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_box_shape_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_capsule_shape_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_concave_polygon_shape_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_cone_twist_joint_get_param**(joint: RID, param: ConeTwistJointParam) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_cone_twist_joint_set_param**(joint: RID, param: ConeTwistJointParam, value: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_convex_polygon_shape_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_custom_shape_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_cylinder_shape_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_end_sync**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_finish**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_flush_queries**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_free_rid**(rid: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_generic_6dof_joint_get_flag**(joint: RID, axis: Vector3.Axis, flag: G6DOFJointAxisFlag) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_generic_6dof_joint_get_param**(joint: RID, axis: Vector3.Axis, param: G6DOFJointAxisParam) virtual const

There is currently no description for this method. Please help us by contributing one!

---

void **_generic_6dof_joint_set_flag**(joint: RID, axis: Vector3.Axis, flag: G6DOFJointAxisFlag, enable: bool) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_generic_6dof_joint_set_param**(joint: RID, axis: Vector3.Axis, param: G6DOFJointAxisParam, value: float) virtual

There is currently no description for this method. Please help us by contributing one!

---

int **_get_process_info**(process_info: ProcessInfo) virtual

There is currently no description for this method. Please help us by contributing one!

RID **_heightmap_shape_create**() virtual

There is currently no description for this method. Please help us by contributing one!

---

bool **_hinge_joint_get_flag**(joint: RID, flag: HingeJointFlag) virtual const

There is currently no description for this method. Please help us by contributing one!

---

float **_hinge_joint_get_param**(joint: RID, param: HingeJointParam) virtual const

There is currently no description for this method. Please help us by contributing one!

---

void **_hinge_joint_set_flag**(joint: RID, flag: HingeJointFlag, enabled: bool) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_hinge_joint_set_param**(joint: RID, param: HingeJointParam, value: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_init**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_is_flushing_queries**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_joint_clear**(joint: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_joint_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_joint_disable_collisions_between_bodies**(joint: RID, disable: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_joint_get_solver_priority**(joint: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

JointType **_joint_get_type**(joint: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_joint_is_disabled_collisions_between_bodies**(joint: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_joint_make_cone_twist**(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_joint_make_generic_6dof**(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_joint_make_hinge**(joint: RID, body_A: RID, hinge_A: Transform3D, body_B: RID, hinge_B: Transform3D) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_joint_make_hinge_simple**(joint: RID, body_A: RID, pivot_A: Vector3, axis_A: Vector3, body_B: RID, pivot_B: Vector3, axis_B: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_joint_make_pin**(joint: RID, body_A: RID, local_A: Vector3, body_B: RID, local_B: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_joint_make_slider**(joint: RID, body_A: RID, local_ref_A: Transform3D, body_B: RID, local_ref_B: Transform3D) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_joint_set_solver_priority**(joint: RID, priority: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_pin_joint_get_local_a**(joint: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_pin_joint_get_local_b**(joint: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_pin_joint_get_param**(joint: RID, param: PinJointParam) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_pin_joint_set_local_a**(joint: RID, local_A: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_pin_joint_set_local_b**(joint: RID, local_B: Vector3) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_pin_joint_set_param**(joint: RID, param: PinJointParam, value: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_separation_ray_shape_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_set_active**(active: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_shape_get_custom_solver_bias**(shape: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Variant **_shape_get_data**(shape: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_shape_get_margin**(shape: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

ShapeType **_shape_get_type**(shape: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_shape_set_custom_solver_bias**(shape: RID, bias: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_shape_set_data**(shape: RID, data: Variant) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_shape_set_margin**(shape: RID, margin: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_slider_joint_get_param**(joint: RID, param: SliderJointParam) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_slider_joint_set_param**(joint: RID, param: SliderJointParam, value: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_add_collision_exception**(body: RID, body_b: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

RID **_soft_body_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

AABB **_soft_body_get_bounds**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Array[RID] **_soft_body_get_collision_exceptions**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_soft_body_get_collision_layer**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_soft_body_get_collision_mask**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_soft_body_get_damping_coefficient**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_soft_body_get_drag_coefficient**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_soft_body_get_linear_stiffness**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Vector3 **_soft_body_get_point_global_position**(body: RID, point_index: int) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

float **_soft_body_get_pressure_coefficient**(body: RID) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_soft_body_get_simulation_precision**(body: RID) virtual const

There is currently no description for this method. Please help us by contributing one!

RID **_soft_body_get_space**(body: RID) virtual const

There is currently no description for this method. Please help us by contributing one!

Variant **_soft_body_get_state**(body: RID, state: BodyState) virtual const

There is currently no description for this method. Please help us by contributing one!

---

float **_soft_body_get_total_mass**(body: RID) virtual const

There is currently no description for this method. Please help us by contributing one!

---

bool **_soft_body_is_point_pinned**(body: RID, point_index: int) virtual const

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_move_point**(body: RID, point_index: int, global_position: Vector3) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_pin_point**(body: RID, point_index: int, pin: bool) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_remove_all_pinned_points**(body: RID) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_remove_collision_exception**(body: RID, body_b: RID) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_set_collision_layer**(body: RID, layer: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_soft_body_set_collision_mask**(body: RID, mask: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_soft_body_set_damping_coefficient**(body: RID, damping_coefficient: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_soft_body_set_drag_coefficient**(body: RID, drag_coefficient: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_soft_body_set_linear_stiffness**(body: RID, linear_stiffness: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_soft_body_set_mesh**(body: RID, mesh: RID) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_soft_body_set_pressure_coefficient**(body: RID, pressure_coefficient: float) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_soft_body_set_ray_pickable**(body: RID, enable: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_soft_body_set_simulation_precision**(body: RID, simulation_precision: int) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_set_space**(body: RID, space: RID) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_set_state**(body: RID, state: BodyState, variant: Variant) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_set_total_mass**(body: RID, total_mass: float) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_set_transform**(body: RID, transform: Transform3D) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_soft_body_update_rendering_server**(body: RID, rendering_server_handler: PhysicsServer3DRenderingServerHandler) virtual

There is currently no description for this method. Please help us by contributing one!

RID **_space_create**() virtual

There is currently no description for this method. Please help us by contributing one!

---

int **_space_get_contact_count**(space: RID) virtual const

There is currently no description for this method. Please help us by contributing one!

PackedVector3Array **_space_get_contacts**(space: RID) virtual const

There is currently no description for this method. Please help us by contributing one!

PhysicsDirectSpaceState3D **_space_get_direct_state**(space: RID) virtual

There is currently no description for this method. Please help us by contributing one!

---

float **_space_get_param**(space: RID, param: SpaceParameter) virtual const

There is currently no description for this method. Please help us by contributing one!

---

bool **_space_is_active**(space: RID) virtual const

There is currently no description for this method. Please help us by contributing one!

---

void **_space_set_active**(space: RID, active: bool) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_space_set_debug_contacts**(space: RID, max_contacts: int) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_space_set_param**(space: RID, param: SpaceParameter, value: float) virtual

There is currently no description for this method. Please help us by contributing one!

RID **_sphere_shape_create**() virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_step**(step: float) virtual

There is currently no description for this method. Please help us by contributing one!

---

void **_sync**() virtual

There is currently no description for this method. Please help us by contributing one!

RID **_world_boundary_shape_create**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **body_test_motion_is_excluding_body**(body: RID) const 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **body_test_motion_is_excluding_object**(object: int) const 🔗

There is currently no description for this method. Please help us by contributing one!

PhysicsServer3DManager

Inherits: Object

A singleton for managing PhysicsServer3D implementations.

Description

PhysicsServer3DManager is the API for registering PhysicsServer3D implementations and for setting the default implementation.

Note: It is not possible to switch physics servers at runtime. This class is only used on startup at the server initialization level, by Godot itself and possibly by GDExtensions.

Methods

void	register_server(name: String, create_callback: Callable)
void	set_default_server(name: String, priority: int)

Method Descriptions

void **register_server**(name: String, create_callback: Callable) 🔗

Register a PhysicsServer3D implementation by passing a `name` and a Callable that returns a PhysicsServer3D object.

---

void **set_default_server**(name: String, priority: int) 🔗

Set the default PhysicsServer3D implementation to the one identified by `name`, if `priority` is greater than the priority of the current default implementation.

PhysicsServer3DRenderingServerHandler

Inherits: Object

A class used to provide PhysicsServer3DExtension._soft_body_update_rendering_server with a rendering handler for soft bodies.

Methods

void	_set_aabb(aabb: AABB) virtual
void	_set_normal(vertex_id: int, normal: Vector3) virtual
void	_set_vertex(vertex_id: int, vertex: Vector3) virtual
void	set_aabb(aabb: AABB)
void	set_normal(vertex_id: int, normal: Vector3)
void	set_vertex(vertex_id: int, vertex: Vector3)

Method Descriptions

void **_set_aabb**(aabb: AABB) virtual

Called by the PhysicsServer3D to set the bounding box for the SoftBody3D.

---

void **_set_normal**(vertex_id: int, normal: Vector3) virtual

Called by the PhysicsServer3D to set the normal for the SoftBody3D vertex at the index specified by `vertex_id`.

Note: The `normal` parameter used to be of type `const void*` prior to Godot 4.2.

---

void **_set_vertex**(vertex_id: int, vertex: Vector3) virtual

Called by the PhysicsServer3D to set the position for the SoftBody3D vertex at the index specified by `vertex_id`.

Note: The `vertex` parameter used to be of type `const void*` prior to Godot 4.2.

---

void **set_aabb**(aabb: AABB)

Sets the bounding box for the SoftBody3D.

---

void **set_normal**(vertex_id: int, normal: Vector3)

Sets the normal for the SoftBody3D vertex at the index specified by `vertex_id`.

---

void **set_vertex**(vertex_id: int, vertex: Vector3)

Sets the position for the SoftBody3D vertex at the index specified by `vertex_id`.

PhysicsShapeQueryParameters2D

Inherits: RefCounted **<** Object

Provides parameters for PhysicsDirectSpaceState2D.intersect_shape.

Description

By changing various properties of this object, such as the shape, you can configure the parameters for PhysicsDirectSpaceState2D.intersect_shape.

Properties

bool	collide_with_areas	`false`
bool	collide_with_bodies	`true`
int	collision_mask	`4294967295`
Array[RID]	exclude	`[]`
float	margin	`0.0`

Vector2	motion	`Vector2(0, 0)`
Resource	shape	
RID	shape_rid	`RID()`
Transform2D	transform	`Transform2D(1, 0, 0, 1, 0, 0)`

Property Descriptions

bool **collide_with_areas** = `false`

- void **set_collide_with_areas**(value: bool)
- bool **is_collide_with_areas_enabled**()

If `true`, the query will take Area2Ds into account.

---

bool **collide_with_bodies** = `true`

- void **set_collide_with_bodies**(value: bool)
- bool **is_collide_with_bodies_enabled**()

If `true`, the query will take PhysicsBody2Ds into account.

---

int **collision_mask** = `4294967295`

- void **set_collision_mask**(value: int)
- int **get_collision_mask**()

The physics layers the query will detect (as a bitmask). By default, all collision layers are detected. See Collision layers and masks in the documentation for more information.

Array[RID] **exclude** = `[]`

- void **set_exclude**(value: Array[RID])
- Array[RID] **get_exclude**()

The list of object RIDs that will be excluded from collisions. Use CollisionObject2D.get_rid to get the RID associated with a CollisionObject2D-derived node.

Note: The returned array is copied and any changes to it will not update the original property value. To update the value you need to modify the returned array, and then assign it to the property again.

---

float **margin** = `0.0`

- void **set_margin**(value: float)
- float **get_margin**()

The collision margin for the shape.

Vector2 **motion** = `Vector2(0, 0)`

- void **set_motion**(value: Vector2)
- Vector2 **get_motion**()

The motion of the shape being queried for.

Resource **shape**

- void **set_shape**(value: Resource)
- Resource **get_shape**()

The Shape2D that will be used for collision/intersection queries. This stores the actual reference which avoids the shape to be released while being used for queries, so always prefer using this over shape_rid.

RID **shape_rid** = `RID()` 🔗

- void **set_shape_rid**(value: RID)
- RID **get_shape_rid**()

The queried shape's RID that will be used for collision/intersection queries. Use this over shape if you want to optimize for performance using the Servers API:

GDScript

```
var shape_rid = PhysicsServer2D.circle_shape_create()
var radius = 64
PhysicsServer2D.shape_set_data(shape_rid, radius)

var params = PhysicsShapeQueryParameters2D.new()
params.shape_rid = shape_rid

# Execute physics queries here...

# Release the shape when done with physics queries.
PhysicsServer2D.free_rid(shape_rid)
```

C#

```
RID shapeRid = PhysicsServer2D.CircleShapeCreate();
int radius = 64;
PhysicsServer2D.ShapeSetData(shapeRid, radius);

var params = new PhysicsShapeQueryParameters2D();
params.ShapeRid = shapeRid;
```

```
// Execute physics queries here...

// Release the shape when done with physics queries.
PhysicsServer2D.FreeRid(shapeRid);
```

Transform2D **transform** = `Transform2D(1, 0, 0, 1, 0, 0)` 🔗

- void **set_transform**(value: Transform2D)
- Transform2D **get_transform**()

The queried shape's transform matrix.

PhysicsShapeQueryParameters3D

Inherits: RefCounted **<** Object

Provides parameters for PhysicsDirectSpaceState3D.intersect_shape.

Description

By changing various properties of this object, such as the shape, you can configure the parameters for PhysicsDirectSpaceState3D.intersect_shape.

Properties

bool	collide_with_areas	`false`
bool	collide_with_bodies	`true`
int	collision_mask	`4294967295`
Array[RID]	exclude	`[]`
float	margin	`0.0`

Vector3	motion	`Vector3(0, 0, 0)`
Resource	shape	
RID	shape_rid	`RID()`
Transform3D	transform	`Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0)`

Property Descriptions

bool **collide_with_areas** = `false` 🔗

- void **set_collide_with_areas**(value: bool)
- bool **is_collide_with_areas_enabled**()

If `true`, the query will take Area3Ds into account.

---

bool **collide_with_bodies** = `true` 🔗

- void **set_collide_with_bodies**(value: bool)
- bool **is_collide_with_bodies_enabled**()

If `true`, the query will take PhysicsBody3Ds into account.

---

int **collision_mask** = `4294967295` 🔗

- void **set_collision_mask**(value: int)

- int **get_collision_mask**()

The physics layers the query will detect (as a bitmask). By default, all collision layers are detected. See Collision layers and masks in the documentation for more information.

Array[RID] **exclude** = `[]`

- void **set_exclude**(value: Array[RID])
- Array[RID] **get_exclude**()

The list of object RIDs that will be excluded from collisions. Use CollisionObject3D.get_rid to get the RID associated with a CollisionObject3D-derived node.

Note: The returned array is copied and any changes to it will not update the original property value. To update the value you need to modify the returned array, and then assign it to the property again.

---

float **margin** = `0.0`

- void **set_margin**(value: float)
- float **get_margin**()

The collision margin for the shape.

Vector3 **motion** = `Vector3(0, 0, 0)`

- void **set_motion**(value: Vector3)
- Vector3 **get_motion**()

The motion of the shape being queried for.

Resource **shape** 🔗

- void **set_shape**(value: Resource)
- Resource **get_shape**()

The Shape3D that will be used for collision/intersection queries. This stores the actual reference which avoids the shape to be released while being used for queries, so always prefer using this over shape_rid.

RID **shape_rid** = `RID()` 🔗

- void **set_shape_rid**(value: RID)
- RID **get_shape_rid**()

The queried shape's RID that will be used for collision/intersection queries. Use this over shape if you want to optimize for performance using the Servers API:

GDScript

```
var shape_rid =
PhysicsServer3D.shape_create(PhysicsServer3D.SHAPE_SPHERE)
var radius = 2.0
PhysicsServer3D.shape_set_data(shape_rid, radius)

var params = PhysicsShapeQueryParameters3D.new()
params.shape_rid = shape_rid

# Execute physics queries here...

# Release the shape when done with physics queries.
PhysicsServer3D.free_rid(shape_rid)
```

C#

```
RID shapeRid =
PhysicsServer3D.ShapeCreate(PhysicsServer3D.ShapeType.Sphere);
```

```
float radius = 2.0f;
PhysicsServer3D.ShapeSetData(shapeRid, radius);

var params = new PhysicsShapeQueryParameters3D();
params.ShapeRid = shapeRid;

// Execute physics queries here...

// Release the shape when done with physics queries.
PhysicsServer3D.FreeRid(shapeRid);
```

Transform3D **transform** = `Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0)` 🔗

- void **set_transform**(value: Transform3D)
- Transform3D **get_transform**()

The queried shape's transform matrix.

PhysicsTestMotionParameters2D

Inherits: RefCounted **<** Object

Provides parameters for PhysicsServer2D.body_test_motion.

Description

By changing various properties of this object, such as the motion, you can configure the parameters for PhysicsServer2D.body_test_motion.

Properties

bool	collide_separation_ray	`false`
Array[RID]	exclude_bodies	`[]`
Array[int]	exclude_objects	`[]`
Transform2D	from	`Transform2D(1, 0, 0, 1, 0, 0)`
float	margin	`0.08`

Vector2	motion	`Vector2(0, 0)`
bool	recovery_as_collision	`false`

Property Descriptions

bool **collide_separation_ray** = `false` 🔗

- void **set_collide_separation_ray_enabled**(value: bool)
- bool **is_collide_separation_ray_enabled**()

If set to `true`, shapes of type PhysicsServer2D.SHAPE_SEPARATION_RAY are used to detect collisions and can stop the motion. Can be useful when snapping to the ground.

If set to `false`, shapes of type PhysicsServer2D.SHAPE_SEPARATION_RAY are only used for separation when overlapping with other bodies. That's the main use for separation ray shapes.

Array[RID] **exclude_bodies** = `[]` 🔗

- void **set_exclude_bodies**(value: Array[RID])
- Array[RID] **get_exclude_bodies**()

Optional array of body RID to exclude from collision. Use CollisionObject2D.get_rid to get the RID associated with a CollisionObject2D-derived node.

Array[int] **exclude_objects** = `[]`

- void **set_exclude_objects**(value: Array[int])
- Array[int] **get_exclude_objects**()

Optional array of object unique instance ID to exclude from collision. See Object.get_instance_id.

Transform2D **from** = `Transform2D(1, 0, 0, 1, 0, 0)`

- void **set_from**(value: Transform2D)
- Transform2D **get_from**()

Transform in global space where the motion should start. Usually set to Node2D.global_transform for the current body's transform.

---

float **margin** = `0.08`

- void **set_margin**(value: float)
- float **get_margin**()

Increases the size of the shapes involved in the collision detection.

Vector2 **motion** = `Vector2(0, 0)`

- void **set_motion**(value: Vector2)
- Vector2 **get_motion**()

Motion vector to define the length and direction of the motion to test.

---

bool **recovery_as_collision** = `false`

- void **set_recovery_as_collision_enabled**(value: bool)
- bool **is_recovery_as_collision_enabled**()

If set to `true`, any depenetration from the recovery phase is reported as a collision; this is used e.g. by CharacterBody2D for improving floor detection during floor snapping.

If set to `false`, only collisions resulting from the motion are reported, which is generally the desired behavior.

PhysicsTestMotionParameters3D

Inherits: RefCounted **<** Object

Provides parameters for PhysicsServer3D.body_test_motion.

Description

By changing various properties of this object, such as the motion, you can configure the parameters for PhysicsServer3D.body_test_motion.

Properties

bool	collide_separation_ray	`false`
Array[RID]	exclude_bodies	`[]`
Array[int]	exclude_objects	`[]`
Transform3D	from	`Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0)`
float	margin	`0.001`

int	max_collisions	`1`
Vector3	motion	`Vector3(0, 0, 0)`
bool	recovery_as_collision	`false`

Property Descriptions

bool **collide_separation_ray** = `false` 🔗

- void **set_collide_separation_ray_enabled**(value: bool)
- bool **is_collide_separation_ray_enabled**()

If set to `true`, shapes of type PhysicsServer3D.SHAPE_SEPARATION_RAY are used to detect collisions and can stop the motion. Can be useful when snapping to the ground.

If set to `false`, shapes of type PhysicsServer3D.SHAPE_SEPARATION_RAY are only used for separation when overlapping with other bodies. That's the main use for separation ray shapes.

Array[RID] **exclude_bodies** = `[]` 🔗

- void **set_exclude_bodies**(value: Array[RID])
- Array[RID] **get_exclude_bodies**()

Optional array of body RID to exclude from collision. Use CollisionObject3D.get_rid to get the RID associated with a

CollisionObject3D-derived node.

Array[int] **exclude_objects** = `[]`

- void **set_exclude_objects**(value: Array[int])
- Array[int] **get_exclude_objects**()

Optional array of object unique instance ID to exclude from collision. See Object.get_instance_id.

Transform3D **from** = `Transform3D(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0)`

- void **set_from**(value: Transform3D)
- Transform3D **get_from**()

Transform in global space where the motion should start. Usually set to Node3D.global_transform for the current body's transform.

---

float **margin** = `0.001`

- void **set_margin**(value: float)
- float **get_margin**()

Increases the size of the shapes involved in the collision detection.

---

int **max_collisions** = `1`

- void **set_max_collisions**(value: int)
- int **get_max_collisions**()

Maximum number of returned collisions, between `1` and `32`. Always returns the deepest detected collisions.

Vector3 **motion** = `Vector3(0, 0, 0)` 🔗

- void **set_motion**(value: Vector3)
- Vector3 **get_motion**()

Motion vector to define the length and direction of the motion to test.

---

bool **recovery_as_collision** = `false` 🔗

- void **set_recovery_as_collision_enabled**(value: bool)
- bool **is_recovery_as_collision_enabled**()

If set to `true`, any depenetration from the recovery phase is reported as a collision; this is used e.g. by CharacterBody3D for improving floor detection during floor snapping.

If set to `false`, only collisions resulting from the motion are reported, which is generally the desired behavior.

PhysicsTestMotionResult2D

Inherits: RefCounted < Object

Describes the motion and collision result from PhysicsServer2D.body_test_motion.

Description

Describes the motion and collision result from PhysicsServer2D.body_test_motion.

Methods

Object	get_collider() const
int	get_collider_id() const
RID	get_collider_rid() const
int	get_collider_shape() const
Vector2	get_collider_velocity() const
float	get_collision_depth() const

int	get_collision_local_shape() const
Vector2	get_collision_normal() const
Vector2	get_collision_point() const
float	get_collision_safe_fraction() const
float	get_collision_unsafe_fraction() const
Vector2	get_remainder() const
Vector2	get_travel() const

Method Descriptions

Object **get_collider**() const 🔗

Returns the colliding body's attached Object, if a collision occurred.

---

int **get_collider_id**() const 🔗

Returns the unique instance ID of the colliding body's attached Object, if a collision occurred. See Object.get_instance_id.

RID **get_collider_rid**() const 🔗

Returns the colliding body's RID used by the PhysicsServer2D, if a collision occurred.

int **get_collider_shape**() const 🔗

Returns the colliding body's shape index, if a collision occurred. See CollisionObject2D.

Vector2 **get_collider_velocity**() const 🔗

Returns the colliding body's velocity, if a collision occurred.

float **get_collision_depth**() const 🔗

Returns the length of overlap along the collision normal, if a collision occurred.

int **get_collision_local_shape**() const 🔗

Returns the moving object's colliding shape, if a collision occurred.

Vector2 **get_collision_normal**() const 🔗

Returns the colliding body's shape's normal at the point of collision, if a collision occurred.

Vector2 **get_collision_point**() const 🔗

Returns the point of collision in global coordinates, if a collision occurred.

float **get_collision_safe_fraction**() const

Returns the maximum fraction of the motion that can occur without a collision, between `0` and `1`.

float **get_collision_unsafe_fraction**() const

Returns the minimum fraction of the motion needed to collide, if a collision occurred, between `0` and `1`.

Vector2 **get_remainder**() const

Returns the moving object's remaining movement vector.

Vector2 **get_travel**() const

Returns the moving object's travel before collision.

PhysicsTestMotionResult3D

Inherits: RefCounted **<** Object

Describes the motion and collision result from PhysicsServer3D.body_test_motion.

Description

Describes the motion and collision result from PhysicsServer3D.body_test_motion.

Methods

Object	get_collider(collision_index: int = 0) const
int	get_collider_id(collision_index: int = 0) const
RID	get_collider_rid(collision_index: int = 0) const
int	get_collider_shape(collision_index: int = 0) const
Vector3	get_collider_velocity(collision_index: int = 0) const
int	get_collision_count() const

float	get_collision_depth(collision_index: int = 0) const
int	get_collision_local_shape(collision_index: int = 0) const
Vector3	get_collision_normal(collision_index: int = 0) const
Vector3	get_collision_point(collision_index: int = 0) const
float	get_collision_safe_fraction() const
float	get_collision_unsafe_fraction() const
Vector3	get_remainder() const
Vector3	get_travel() const

Method Descriptions

Object **get_collider**(collision_index: int = 0) const 🔗

Returns the colliding body's attached Object given a collision index (the deepest collision by default), if a collision occurred.

int **get_collider_id**(collision_index: int = 0) const 🔗

Returns the unique instance ID of the colliding body's attached Object given a collision index (the deepest collision by default), if a collision occurred. See Object.get_instance_id.

RID **get_collider_rid**(collision_index: int = 0) const

Returns the colliding body's RID used by the PhysicsServer3D given a collision index (the deepest collision by default), if a collision occurred.

---

int **get_collider_shape**(collision_index: int = 0) const

Returns the colliding body's shape index given a collision index (the deepest collision by default), if a collision occurred. See CollisionObject3D.

Vector3 **get_collider_velocity**(collision_index: int = 0) const

Returns the colliding body's velocity given a collision index (the deepest collision by default), if a collision occurred.

---

int **get_collision_count**() const

Returns the number of detected collisions.

---

float **get_collision_depth**(collision_index: int = 0) const

Returns the length of overlap along the collision normal given a collision index (the deepest collision by default), if a collision occurred.

---

int **get_collision_local_shape**(collision_index: int = 0) const

Returns the moving object's colliding shape given a collision index (the deepest collision by default), if a collision occurred.

Vector3 **get_collision_normal**(collision_index: int = 0) const

Returns the colliding body's shape's normal at the point of collision given a collision index (the deepest collision by default), if a collision occurred.

Vector3 **get_collision_point**(collision_index: int = 0) const

Returns the point of collision in global coordinates given a collision index (the deepest collision by default), if a collision occurred.

---

float **get_collision_safe_fraction**() const

Returns the maximum fraction of the motion that can occur without a collision, between `0` and `1`.

---

float **get_collision_unsafe_fraction**() const

Returns the minimum fraction of the motion needed to collide, if a collision occurred, between `0` and `1`.

Vector3 **get_remainder**() const

Returns the moving object's remaining movement vector.

Vector3 **get_travel**() const

Returns the moving object's travel before collision.

ProjectSettings

Inherits: Object

Stores globally-accessible variables.

Description

Stores variables that can be accessed from everywhere. Use get_setting, set_setting or has_setting to access them. Variables stored in `project.godot` are also loaded into **ProjectSettings**, making this object very useful for reading custom game configuration options.

When naming a Project Settings property, use the full path to the setting including the category. For example, `"application/config/name"` for the project name. Category and property names can be viewed in the Project Settings dialog.

Feature tags: Project settings can be overridden for specific platforms and configurations (debug, release, ...) using feature tags.

Overriding: Any project setting can be overridden by creating a file named `override.cfg` in the project's root directory. This can also be used in exported projects by placing this file in the same directory as the project binary. Overriding will still take the base project settings' feature tags in account. Therefore, make sure to *also* override the setting with the desired feature tags if you want them to override base project settings on all platforms and configurations.

Tutorials

- 3D Physics Tests Demo [https://godotengine.org/asset-library/asset/2747]
- 3D Platformer Demo [https://godotengine.org/asset-library/asset/2748]
- Operating System Testing Demo [https://godotengine.org/asset-library/asset/2789]

Properties

bool	animation/warnings/check_angle_interpolation_type_conflicting
bool	animation/warnings/check_invalid_track_paths
Color	application/boot_splash/bg_color
bool	application/boot_splash/fullsize
String	application/boot_splash/image
int	application/boot_splash/minimum_display_time

bool	application/boot_splash/show_image
bool	application/boot_splash/use_filter
bool	application/config/auto_accept_quit
String	application/config/custom_user_dir_name
String	application/config/description
String	application/config/icon
String	application/config/macos_native_icon
String	application/config/name
Dictionary	application/config/name_localized
String	application/config/project_settings_override
bool	application/config/quit_on_go_back
bool	application/config/use_custom_user_dir
bool	application/config/use_hidden_project_data_directory
String	application/config/version
String	application/config/windows_native_icon
bool	application/run/delta_smoothing
bool	application/run/disable_stderr
bool	application/run/disable_stdout
bool	application/run/enable_alt_space_menu

bool	application/run/flush_stdout_on_print
bool	application/run/flush_stdout_on_print.debug
int	application/run/frame_delay_msec
bool	application/run/low_processor_mode
int	application/run/low_processor_mode_sleep_usec
String	application/run/main_loop_type
String	application/run/main_scene
int	application/run/max_fps
bool	application/run/print_header
float	audio/buses/channel_disable_threshold_db
float	audio/buses/channel_disable_time
String	audio/buses/default_bus_layout
String	audio/driver/driver
bool	audio/driver/enable_input
int	audio/driver/mix_rate
int	audio/driver/mix_rate.web
int	audio/driver/output_latency
int	audio/driver/output_latency.web
float	audio/general/2d_panning_strength

float	audio/general/3d_panning_strength
int	audio/general/default_playback_type
int	audio/general/default_playback_type.web
bool	audio/general/ios/mix_with_others
int	audio/general/ios/session_category
bool	audio/general/text_to_speech
int	audio/video/video_delay_compensation_ms
bool	collada/use_ambient
int	compression/formats/gzip/compression_level
int	compression/formats/zlib/compression_level
int	compression/formats/zstd/compression_level
bool	compression/formats/zstd/long_distance_matching
int	compression/formats/zstd/window_log_size
Color	debug/canvas_items/debug_redraw_color
float	debug/canvas_items/debug_redraw_time
bool	debug/file_logging/enable_file_logging
bool	debug/file_logging/enable_file_logging.pc
String	debug/file_logging/log_path
int	debug/file_logging/max_log_files

int	debug/gdscript/warnings/assert_always_false
int	debug/gdscript/warnings/assert_always_true
int	debug/gdscript/warnings/confusable_capture_reassignment
int	debug/gdscript/warnings/confusable_identifier
int	debug/gdscript/warnings/confusable_local_declaration
int	debug/gdscript/warnings/confusable_local_usage
int	debug/gdscript/warnings/constant_used_as_function
int	debug/gdscript/warnings/deprecated_keyword
int	debug/gdscript/warnings/empty_file
bool	debug/gdscript/warnings/enable
int	debug/gdscript/warnings/enum_variable_without_default
bool	debug/gdscript/warnings/exclude_addons
int	debug/gdscript/warnings/function_used_as_property
int	debug/gdscript/warnings/get_node_default_without_onready
int	debug/gdscript/warnings/incompatible_ternary
int	debug/gdscript/warnings/inference_on_variant
int	debug/gdscript/warnings/inferred_declaration
int	debug/gdscript/warnings/int_as_enum_without_cast
int	debug/gdscript/warnings/int_as_enum_without_match

int	debug/gdscript/warnings/integer_division
int	debug/gdscript/warnings/narrowing_conversion
int	debug/gdscript/warnings/native_method_override
int	debug/gdscript/warnings/onready_with_export
int	debug/gdscript/warnings/property_used_as_function
int	debug/gdscript/warnings/redundant_await
int	debug/gdscript/warnings/redundant_static_unload
bool	debug/gdscript/warnings/renamed_in_godot_4_hint
int	debug/gdscript/warnings/return_value_discarded
int	debug/gdscript/warnings/shadowed_global_identifier
int	debug/gdscript/warnings/shadowed_variable
int	debug/gdscript/warnings/shadowed_variable_base_class
int	debug/gdscript/warnings/standalone_expression
int	debug/gdscript/warnings/standalone_ternary
int	debug/gdscript/warnings/static_called_on_instance
int	debug/gdscript/warnings/unassigned_variable
int	debug/gdscript/warnings/unassigned_variable_op_assign
int	debug/gdscript/warnings/unreachable_code
int	debug/gdscript/warnings/unreachable_pattern

int	debug/gdscript/warnings/unsafe_call_argument
int	debug/gdscript/warnings/unsafe_cast
int	debug/gdscript/warnings/unsafe_method_access
int	debug/gdscript/warnings/unsafe_property_access
int	debug/gdscript/warnings/unsafe_void_return
int	debug/gdscript/warnings/untyped_declaration
int	debug/gdscript/warnings/unused_local_constant
int	debug/gdscript/warnings/unused_parameter
int	debug/gdscript/warnings/unused_private_class_variable
int	debug/gdscript/warnings/unused_signal
int	debug/gdscript/warnings/unused_variable
String	debug/settings/crash_handler/message
String	debug/settings/crash_handler/message.editor
int	debug/settings/gdscript/max_call_stack
int	debug/settings/profiler/max_functions
int	debug/settings/profiler/max_timestamp_query_elements
bool	debug/settings/stdout/print_fps
bool	debug/settings/stdout/print_gpu_profile

bool	debug/settings/stdout/verbose_stdout
bool	debug/shader_language/warnings/device_limit_exceeded
bool	debug/shader_language/warnings/enable
bool	debug/shader_language/warnings/float_comparison
bool	debug/shader_language/warnings/formatting_error
bool	debug/shader_language/warnings/magic_position_write
bool	debug/shader_language/warnings/treat_warnings_as_errors
bool	debug/shader_language/warnings/unused_constant
bool	debug/shader_language/warnings/unused_function
bool	debug/shader_language/warnings/unused_local_variable
bool	debug/shader_language/warnings/unused_struct
bool	debug/shader_language/warnings/unused_uniform
bool	debug/shader_language/warnings/unused_varying
Color	debug/shapes/avoidance/agents_radius_color
bool	debug/shapes/avoidance/enable_agents_radius
bool	debug/shapes/avoidance/enable_obstacles_radius
bool	debug/shapes/avoidance/enable_obstacles_static
Color	debug/shapes/avoidance/obstacles_radius_color
Color	debug/shapes/avoidance/obstacles_static_edge_pushin_color

Color	debug/shapes/avoidance/obstacles_static_edge_pushout_color
Color	debug/shapes/avoidance/obstacles_static_face_pushin_color
Color	debug/shapes/avoidance/obstacles_static_face_pushout_color
Color	debug/shapes/collision/contact_color
bool	debug/shapes/collision/draw_2d_outlines
int	debug/shapes/collision/max_contacts_displayed
Color	debug/shapes/collision/shape_color
Color	debug/shapes/navigation/agent_path_color
float	debug/shapes/navigation/agent_path_point_size
Color	debug/shapes/navigation/edge_connection_color
bool	debug/shapes/navigation/enable_agent_paths
bool	debug/shapes/navigation/enable_agent_paths_xray
bool	debug/shapes/navigation/enable_edge_connections
bool	debug/shapes/navigation/enable_edge_connections_xray
bool	debug/shapes/navigation/enable_edge_lines
bool	debug/shapes/navigation/enable_edge_lines_xray
bool	debug/shapes/navigation/enable_geometry_face_random_color
bool	debug/shapes/navigation/enable_link_connections
bool	debug/shapes/navigation/enable_link_connections_xray

Color	debug/shapes/navigation/geometry_edge_color
Color	debug/shapes/navigation/geometry_edge_disabled_color
Color	debug/shapes/navigation/geometry_face_color
Color	debug/shapes/navigation/geometry_face_disabled_color
Color	debug/shapes/navigation/link_connection_color
Color	debug/shapes/navigation/link_connection_disabled_color
Color	debug/shapes/paths/geometry_color
float	debug/shapes/paths/geometry_width
String	display/display_server/driver
String	display/display_server/driver.android
String	display/display_server/driver.ios
String	display/display_server/driver.linuxbsd
String	display/display_server/driver.macos
String	display/display_server/driver.windows
String	display/mouse_cursor/custom_image
Vector2	display/mouse_cursor/custom_image_hotspot
Vector2	display/mouse_cursor/tooltip_position_offset
bool	display/window/dpi/allow_hidpi
bool	display/window/energy_saving/keep_screen_on

bool	display/window/energy_saving/keep_screen_on.editor_hint
int	display/window/handheld/orientation
bool	display/window/ios/allow_high_refresh_rate
bool	display/window/ios/hide_home_indicator
bool	display/window/ios/hide_status_bar
bool	display/window/ios/suppress_ui_gesture
bool	display/window/per_pixel_transparency/allowed
bool	display/window/size/always_on_top
bool	display/window/size/borderless
bool	display/window/size/extend_to_title
Vector2i	display/window/size/initial_position
int	display/window/size/initial_position_type
int	display/window/size/initial_screen
int	display/window/size/mode
bool	display/window/size/no_focus
bool	display/window/size/resizable
bool	display/window/size/transparent
int	display/window/size/viewport_height
int	display/window/size/viewport_width

int	display/window/size/window_height_override
int	display/window/size/window_width_override
String	display/window/stretch/aspect
String	display/window/stretch/mode
float	display/window/stretch/scale
String	display/window/stretch/scale_mode
bool	display/window/subwindows/embed_subwindows
int	display/window/vsync/vsync_mode
String	dotnet/project/assembly_name
int	dotnet/project/assembly_reload_attempts
String	dotnet/project/solution_directory
bool	editor/export/convert_text_resources_to_binary
int	editor/import/atlas_max_width
bool	editor/import/reimport_missing_imported_files
bool	editor/import/use_multiple_threads
bool	editor/movie_writer/disable_vsync
int	editor/movie_writer/fps
int	editor/movie_writer/mix_rate
float	editor/movie_writer/mjpeg_quality

String	editor/movie_writer/movie_file
int	editor/movie_writer/speaker_mode
String	editor/naming/default_signal_callback_name
String	editor/naming/default_signal_callback_to_self_name
int	editor/naming/node_name_casing
int	editor/naming/node_name_num_separator
int	editor/naming/scene_name_casing
int	editor/naming/script_name_casing
String	editor/run/main_run_args
PackedStringArray	editor/script/search_in_file_extensions
String	editor/script/templates_search_path
bool	editor/version_control/autoload_on_startup
String	editor/version_control/plugin_name
bool	filesystem/import/blender/enabled
bool	filesystem/import/blender/enabled.android
bool	filesystem/import/blender/enabled.web
bool	filesystem/import/fbx2gltf/enabled
bool	filesystem/import/fbx2gltf/enabled.android
bool	filesystem/import/fbx2gltf/enabled.web

int	gui/common/default_scroll_deadzone
bool	gui/common/snap_controls_to_pixels
bool	gui/common/swap_cancel_ok
int	gui/common/text_edit_undo_stack_max_size
bool	gui/fonts/dynamic_fonts/use_oversampling
String	gui/theme/custom
String	gui/theme/custom_font
int	gui/theme/default_font_antialiasing
bool	gui/theme/default_font_generate_mipmaps
int	gui/theme/default_font_hinting
bool	gui/theme/default_font_multichannel_signed_distance_field
int	gui/theme/default_font_subpixel_positioning
float	gui/theme/default_theme_scale
int	gui/theme/lcd_subpixel_layout
float	gui/timers/button_shortcut_feedback_highlight_time
int	gui/timers/incremental_search_max_interval_msec
float	gui/timers/text_edit_idle_detect_sec
float	gui/timers/tooltip_delay_sec
float	gui/timers/tooltip_delay_sec.editor_hint

Dictionary	input/ui_accept
Dictionary	input/ui_cancel
Dictionary	input/ui_copy
Dictionary	input/ui_cut
Dictionary	input/ui_down
Dictionary	input/ui_end
Dictionary	input/ui_filedialog_refresh
Dictionary	input/ui_filedialog_show_hidden
Dictionary	input/ui_filedialog_up_one_level
Dictionary	input/ui_focus_next
Dictionary	input/ui_focus_prev
Dictionary	input/ui_graph_delete
Dictionary	input/ui_graph_duplicate
Dictionary	input/ui_home
Dictionary	input/ui_left
Dictionary	input/ui_menu
Dictionary	input/ui_page_down
Dictionary	input/ui_page_up
Dictionary	input/ui_paste

Dictionary	input/ui_redo
Dictionary	input/ui_right
Dictionary	input/ui_select
Dictionary	input/ui_swap_input_direction
Dictionary	input/ui_text_add_selection_for_next_occurrence
Dictionary	input/ui_text_backspace
Dictionary	input/ui_text_backspace_all_to_left
Dictionary	input/ui_text_backspace_all_to_left.macos
Dictionary	input/ui_text_backspace_word
Dictionary	input/ui_text_backspace_word.macos
Dictionary	input/ui_text_caret_add_above
Dictionary	input/ui_text_caret_add_above.macos
Dictionary	input/ui_text_caret_add_below
Dictionary	input/ui_text_caret_add_below.macos
Dictionary	input/ui_text_caret_document_end
Dictionary	input/ui_text_caret_document_end.macos
Dictionary	input/ui_text_caret_document_start
Dictionary	input/ui_text_caret_document_start.macos
Dictionary	input/ui_text_caret_down

Dictionary	input/ui_text_caret_left
Dictionary	input/ui_text_caret_line_end
Dictionary	input/ui_text_caret_line_end.macos
Dictionary	input/ui_text_caret_line_start
Dictionary	input/ui_text_caret_line_start.macos
Dictionary	input/ui_text_caret_page_down
Dictionary	input/ui_text_caret_page_up
Dictionary	input/ui_text_caret_right
Dictionary	input/ui_text_caret_up
Dictionary	input/ui_text_caret_word_left
Dictionary	input/ui_text_caret_word_left.macos
Dictionary	input/ui_text_caret_word_right
Dictionary	input/ui_text_caret_word_right.macos
Dictionary	input/ui_text_clear_carets_and_selection
Dictionary	input/ui_text_completion_accept
Dictionary	input/ui_text_completion_query
Dictionary	input/ui_text_completion_replace
Dictionary	input/ui_text_dedent
Dictionary	input/ui_text_delete

Dictionary	input/ui_text_delete_all_to_right
Dictionary	input/ui_text_delete_all_to_right.macos
Dictionary	input/ui_text_delete_word
Dictionary	input/ui_text_delete_word.macos
Dictionary	input/ui_text_indent
Dictionary	input/ui_text_newline
Dictionary	input/ui_text_newline_above
Dictionary	input/ui_text_newline_blank
Dictionary	input/ui_text_scroll_down
Dictionary	input/ui_text_scroll_down.macos
Dictionary	input/ui_text_scroll_up
Dictionary	input/ui_text_scroll_up.macos
Dictionary	input/ui_text_select_all
Dictionary	input/ui_text_select_word_under_caret
Dictionary	input/ui_text_select_word_under_caret.macos
Dictionary	input/ui_text_skip_selection_for_next_occurrence
Dictionary	input/ui_text_submit
Dictionary	input/ui_text_toggle_insert_mode
Dictionary	input/ui_undo

Dictionary	input/ui_up
bool	input_devices/buffering/agile_event_flushing
bool	input_devices/compatibility/legacy_just_pressed_behavior
String	input_devices/pen_tablet/driver
String	input_devices/pen_tablet/driver.windows
bool	input_devices/pointing/android/enable_long_press_as_right_click
bool	input_devices/pointing/android/enable_pan_and_scale_gestures
int	input_devices/pointing/android/rotary_input_scroll_axis
bool	input_devices/pointing/emulate_mouse_from_touch
bool	input_devices/pointing/emulate_touch_from_mouse
String	internationalization/locale/fallback
bool	internationalization/locale/include_text_server_data
String	internationalization/locale/test
bool	internationalization/pseudolocalization/double_vowels
float	internationalization/pseudolocalization/expansion_ratio
bool	internationalization/pseudolocalization/fake_bidi
bool	internationalization/pseudolocalization/override
String	internationalization/pseudolocalization/prefix
bool	internationalization/pseudolocalization/replace_with_accents

bool	internationalization/pseudolocalization/skip_placeholders
String	internationalization/pseudolocalization/suffix
bool	internationalization/pseudolocalization/use_pseudolocalization
bool	internationalization/rendering/force_right_to_left_layout_direction
bool	internationalization/rendering/root_node_auto_translate
int	internationalization/rendering/root_node_layout_direction
String	internationalization/rendering/text_driver
String	layer_names/2d_navigation/layer_1
String	layer_names/2d_navigation/layer_2
String	layer_names/2d_navigation/layer_3
String	layer_names/2d_navigation/layer_4
String	layer_names/2d_navigation/layer_5
String	layer_names/2d_navigation/layer_6
String	layer_names/2d_navigation/layer_7
String	layer_names/2d_navigation/layer_8
String	layer_names/2d_navigation/layer_9
String	layer_names/2d_navigation/layer_10
String	layer_names/2d_navigation/layer_11
String	layer_names/2d_navigation/layer_12

String	layer_names/2d_navigation/layer_13
String	layer_names/2d_navigation/layer_14
String	layer_names/2d_navigation/layer_15
String	layer_names/2d_navigation/layer_16
String	layer_names/2d_navigation/layer_17
String	layer_names/2d_navigation/layer_18
String	layer_names/2d_navigation/layer_19
String	layer_names/2d_navigation/layer_20
String	layer_names/2d_navigation/layer_21
String	layer_names/2d_navigation/layer_22
String	layer_names/2d_navigation/layer_23
String	layer_names/2d_navigation/layer_24
String	layer_names/2d_navigation/layer_25
String	layer_names/2d_navigation/layer_26
String	layer_names/2d_navigation/layer_27
String	layer_names/2d_navigation/layer_28
String	layer_names/2d_navigation/layer_29
String	layer_names/2d_navigation/layer_30
String	layer_names/2d_navigation/layer_31

String	layer_names/2d_navigation/layer_32
String	layer_names/2d_physics/layer_1
String	layer_names/2d_physics/layer_2
String	layer_names/2d_physics/layer_3
String	layer_names/2d_physics/layer_4
String	layer_names/2d_physics/layer_5
String	layer_names/2d_physics/layer_6
String	layer_names/2d_physics/layer_7
String	layer_names/2d_physics/layer_8
String	layer_names/2d_physics/layer_9
String	layer_names/2d_physics/layer_10
String	layer_names/2d_physics/layer_11
String	layer_names/2d_physics/layer_12
String	layer_names/2d_physics/layer_13
String	layer_names/2d_physics/layer_14
String	layer_names/2d_physics/layer_15
String	layer_names/2d_physics/layer_16
String	layer_names/2d_physics/layer_17
String	layer_names/2d_physics/layer_18

String	layer_names/2d_physics/layer_19
String	layer_names/2d_physics/layer_20
String	layer_names/2d_physics/layer_21
String	layer_names/2d_physics/layer_22
String	layer_names/2d_physics/layer_23
String	layer_names/2d_physics/layer_24
String	layer_names/2d_physics/layer_25
String	layer_names/2d_physics/layer_26
String	layer_names/2d_physics/layer_27
String	layer_names/2d_physics/layer_28
String	layer_names/2d_physics/layer_29
String	layer_names/2d_physics/layer_30
String	layer_names/2d_physics/layer_31
String	layer_names/2d_physics/layer_32
String	layer_names/2d_render/layer_1
String	layer_names/2d_render/layer_2
String	layer_names/2d_render/layer_3
String	layer_names/2d_render/layer_4
String	layer_names/2d_render/layer_5

String	layer_names/2d_render/layer_6
String	layer_names/2d_render/layer_7
String	layer_names/2d_render/layer_8
String	layer_names/2d_render/layer_9
String	layer_names/2d_render/layer_10
String	layer_names/2d_render/layer_11
String	layer_names/2d_render/layer_12
String	layer_names/2d_render/layer_13
String	layer_names/2d_render/layer_14
String	layer_names/2d_render/layer_15
String	layer_names/2d_render/layer_16
String	layer_names/2d_render/layer_17
String	layer_names/2d_render/layer_18
String	layer_names/2d_render/layer_19
String	layer_names/2d_render/layer_20
String	layer_names/3d_navigation/layer_1
String	layer_names/3d_navigation/layer_2
String	layer_names/3d_navigation/layer_3
String	layer_names/3d_navigation/layer_4

String	layer_names/3d_navigation/layer_5
String	layer_names/3d_navigation/layer_6
String	layer_names/3d_navigation/layer_7
String	layer_names/3d_navigation/layer_8
String	layer_names/3d_navigation/layer_9
String	layer_names/3d_navigation/layer_10
String	layer_names/3d_navigation/layer_11
String	layer_names/3d_navigation/layer_12
String	layer_names/3d_navigation/layer_13
String	layer_names/3d_navigation/layer_14
String	layer_names/3d_navigation/layer_15
String	layer_names/3d_navigation/layer_16
String	layer_names/3d_navigation/layer_17
String	layer_names/3d_navigation/layer_18
String	layer_names/3d_navigation/layer_19
String	layer_names/3d_navigation/layer_20
String	layer_names/3d_navigation/layer_21
String	layer_names/3d_navigation/layer_22
String	layer_names/3d_navigation/layer_23

String	layer_names/3d_navigation/layer_24
String	layer_names/3d_navigation/layer_25
String	layer_names/3d_navigation/layer_26
String	layer_names/3d_navigation/layer_27
String	layer_names/3d_navigation/layer_28
String	layer_names/3d_navigation/layer_29
String	layer_names/3d_navigation/layer_30
String	layer_names/3d_navigation/layer_31
String	layer_names/3d_navigation/layer_32
String	layer_names/3d_physics/layer_1
String	layer_names/3d_physics/layer_2
String	layer_names/3d_physics/layer_3
String	layer_names/3d_physics/layer_4
String	layer_names/3d_physics/layer_5
String	layer_names/3d_physics/layer_6
String	layer_names/3d_physics/layer_7
String	layer_names/3d_physics/layer_8
String	layer_names/3d_physics/layer_9
String	layer_names/3d_physics/layer_10

String	layer_names/3d_physics/layer_11
String	layer_names/3d_physics/layer_12
String	layer_names/3d_physics/layer_13
String	layer_names/3d_physics/layer_14
String	layer_names/3d_physics/layer_15
String	layer_names/3d_physics/layer_16
String	layer_names/3d_physics/layer_17
String	layer_names/3d_physics/layer_18
String	layer_names/3d_physics/layer_19
String	layer_names/3d_physics/layer_20
String	layer_names/3d_physics/layer_21
String	layer_names/3d_physics/layer_22
String	layer_names/3d_physics/layer_23
String	layer_names/3d_physics/layer_24
String	layer_names/3d_physics/layer_25
String	layer_names/3d_physics/layer_26
String	layer_names/3d_physics/layer_27
String	layer_names/3d_physics/layer_28
String	layer_names/3d_physics/layer_29

String	layer_names/3d_physics/layer_30
String	layer_names/3d_physics/layer_31
String	layer_names/3d_physics/layer_32
String	layer_names/3d_render/layer_1
String	layer_names/3d_render/layer_2
String	layer_names/3d_render/layer_3
String	layer_names/3d_render/layer_4
String	layer_names/3d_render/layer_5
String	layer_names/3d_render/layer_6
String	layer_names/3d_render/layer_7
String	layer_names/3d_render/layer_8
String	layer_names/3d_render/layer_9
String	layer_names/3d_render/layer_10
String	layer_names/3d_render/layer_11
String	layer_names/3d_render/layer_12
String	layer_names/3d_render/layer_13
String	layer_names/3d_render/layer_14
String	layer_names/3d_render/layer_15
String	layer_names/3d_render/layer_16

String	layer_names/3d_render/layer_17
String	layer_names/3d_render/layer_18
String	layer_names/3d_render/layer_19
String	layer_names/3d_render/layer_20
String	layer_names/avoidance/layer_1
String	layer_names/avoidance/layer_2
String	layer_names/avoidance/layer_3
String	layer_names/avoidance/layer_4
String	layer_names/avoidance/layer_5
String	layer_names/avoidance/layer_6
String	layer_names/avoidance/layer_7
String	layer_names/avoidance/layer_8
String	layer_names/avoidance/layer_9
String	layer_names/avoidance/layer_10
String	layer_names/avoidance/layer_11
String	layer_names/avoidance/layer_12
String	layer_names/avoidance/layer_13
String	layer_names/avoidance/layer_14
String	layer_names/avoidance/layer_15

String	layer_names/avoidance/layer_16
String	layer_names/avoidance/layer_17
String	layer_names/avoidance/layer_18
String	layer_names/avoidance/layer_19
String	layer_names/avoidance/layer_20
String	layer_names/avoidance/layer_21
String	layer_names/avoidance/layer_22
String	layer_names/avoidance/layer_23
String	layer_names/avoidance/layer_24
String	layer_names/avoidance/layer_25
String	layer_names/avoidance/layer_26
String	layer_names/avoidance/layer_27
String	layer_names/avoidance/layer_28
String	layer_names/avoidance/layer_29
String	layer_names/avoidance/layer_30
String	layer_names/avoidance/layer_31
String	layer_names/avoidance/layer_32
int	memory/limits/message_queue/max_size_mb
float	navigation/2d/default_cell_size

float	navigation/2d/default_edge_connection_margin
float	navigation/2d/default_link_connection_radius
bool	navigation/2d/use_edge_connections
float	navigation/3d/default_cell_height
float	navigation/3d/default_cell_size
float	navigation/3d/default_edge_connection_margin
float	navigation/3d/default_link_connection_radius
Vector3	navigation/3d/default_up
float	navigation/3d/merge_rasterizer_cell_scale
bool	navigation/3d/use_edge_connections
bool	navigation/avoidance/thread_model/avoidance_use_high_priority_threads
bool	navigation/avoidance/thread_model/avoidance_use_multiple_threads
bool	navigation/baking/thread_model/baking_use_high_priority_threads
bool	navigation/baking/thread_model/baking_use_multiple_threads
bool	navigation/baking/use_crash_prevention_checks
int	network/limits/debugger/max_chars_per_second
int	network/limits/debugger/max_errors_per_second
int	network/limits/debugger/max_queued_messages
int	network/limits/debugger/max_warnings_per_second

int	network/limits/packet_peer_stream/max_buffer_po2
int	network/limits/tcp/connect_timeout_seconds
int	network/limits/webrtc/max_channel_in_buffer_kb
String	network/tls/certificate_bundle_override
float	physics/2d/default_angular_damp
float	physics/2d/default_gravity
Vector2	physics/2d/default_gravity_vector
float	physics/2d/default_linear_damp
String	physics/2d/physics_engine
bool	physics/2d/run_on_separate_thread
float	physics/2d/sleep_threshold_angular
float	physics/2d/sleep_threshold_linear
float	physics/2d/solver/contact_max_allowed_penetration
float	physics/2d/solver/contact_max_separation
float	physics/2d/solver/contact_recycle_radius
float	physics/2d/solver/default_constraint_bias
float	physics/2d/solver/default_contact_bias
int	physics/2d/solver/solver_iterations
float	physics/2d/time_before_sleep

float	physics/3d/default_angular_damp
float	physics/3d/default_gravity
Vector3	physics/3d/default_gravity_vector
float	physics/3d/default_linear_damp
String	physics/3d/physics_engine
bool	physics/3d/run_on_separate_thread
float	physics/3d/sleep_threshold_angular
float	physics/3d/sleep_threshold_linear
float	physics/3d/solver/contact_max_allowed_penetration
float	physics/3d/solver/contact_max_separation
float	physics/3d/solver/contact_recycle_radius
float	physics/3d/solver/default_contact_bias
int	physics/3d/solver/solver_iterations
float	physics/3d/time_before_sleep
bool	physics/common/enable_object_picking
int	physics/common/max_physics_steps_per_frame
bool	physics/common/physics_interpolation
float	physics/common/physics_jitter_fix
int	physics/common/physics_ticks_per_second

int	rendering/2d/sdf/oversize
int	rendering/2d/sdf/scale
int	rendering/2d/shadow_atlas/size
bool	rendering/2d/snap/snap_2d_transforms_to_pixel
bool	rendering/2d/snap/snap_2d_vertices_to_pixel
int	rendering/anti_aliasing/quality/msaa_2d
int	rendering/anti_aliasing/quality/msaa_3d
int	rendering/anti_aliasing/quality/screen_space_aa
bool	rendering/anti_aliasing/quality/use_debanding
bool	rendering/anti_aliasing/quality/use_taa
float	rendering/anti_aliasing/screen_space_roughness_limiter/amount
bool	rendering/anti_aliasing/screen_space_roughness_limiter/enabled
float	rendering/anti_aliasing/screen_space_roughness_limiter/limit
int	rendering/camera/depth_of_field/depth_of_field_bokeh_quality
int	rendering/camera/depth_of_field/depth_of_field_bokeh_shape
bool	rendering/camera/depth_of_field/depth_of_field_use_jitter
String	rendering/driver/depth_prepass/disable_for_vendors
bool	rendering/driver/depth_prepass/enable
int	rendering/driver/threads/thread_model

Color	rendering/environment/defaults/default_clear_color
String	rendering/environment/defaults/default_environment
int	rendering/environment/glow/upscale_mode
int	rendering/environment/glow/upscale_mode.mobile
int	rendering/environment/screen_space_reflection/roughness_quality
float	rendering/environment/ssao/adaptive_target
int	rendering/environment/ssao/blur_passes
float	rendering/environment/ssao/fadeout_from
float	rendering/environment/ssao/fadeout_to
bool	rendering/environment/ssao/half_size
int	rendering/environment/ssao/quality
float	rendering/environment/ssil/adaptive_target
int	rendering/environment/ssil/blur_passes
float	rendering/environment/ssil/fadeout_from
float	rendering/environment/ssil/fadeout_to
bool	rendering/environment/ssil/half_size
int	rendering/environment/ssil/quality
float	rendering/environment/subsurface_scattering/subsurface_scattering_depth_scale
int	rendering/environment/subsurface_scattering/subsurface_scattering_quality

float	rendering/environment/subsurface_scattering/subsurface_scattering_scale
int	rendering/environment/volumetric_fog/use_filter
int	rendering/environment/volumetric_fog/volume_depth
int	rendering/environment/volumetric_fog/volume_size
String	rendering/gl_compatibility/driver
String	rendering/gl_compatibility/driver.android
String	rendering/gl_compatibility/driver.ios
String	rendering/gl_compatibility/driver.linuxbsd
String	rendering/gl_compatibility/driver.macos
String	rendering/gl_compatibility/driver.web
String	rendering/gl_compatibility/driver.windows
bool	rendering/gl_compatibility/fallback_to_angle
bool	rendering/gl_compatibility/fallback_to_gles
bool	rendering/gl_compatibility/fallback_to_native
Array	rendering/gl_compatibility/force_angle_on_devices
int	rendering/gl_compatibility/item_buffer_size
bool	rendering/gl_compatibility/nvidia_disable_threaded_optimization
bool	rendering/global_illumination/gi/use_half_resolution
int	rendering/global_illumination/sdfgi/frames_to_converge

int	rendering/global_illumination/sdfgi/frames_to_update_lights
int	rendering/global_illumination/sdfgi/probe_ray_count
int	rendering/global_illumination/voxel_gi/quality
int	rendering/lightmapping/bake_performance/max_rays_per_pass
int	rendering/lightmapping/bake_performance/max_rays_per_probe_pass
int	rendering/lightmapping/bake_performance/region_size
int	rendering/lightmapping/bake_quality/high_quality_probe_ray_count
int	rendering/lightmapping/bake_quality/high_quality_ray_count
int	rendering/lightmapping/bake_quality/low_quality_probe_ray_count
int	rendering/lightmapping/bake_quality/low_quality_ray_count
int	rendering/lightmapping/bake_quality/medium_quality_probe_ray_count
int	rendering/lightmapping/bake_quality/medium_quality_ray_count
int	rendering/lightmapping/bake_quality/ultra_quality_probe_ray_count
int	rendering/lightmapping/bake_quality/ultra_quality_ray_count
int	rendering/lightmapping/denoising/denoiser
float	rendering/lightmapping/primitive_meshes/texel_size
float	rendering/lightmapping/probe_capture/update_speed
bool	rendering/lights_and_shadows/directional_shadow/16_bits
int	rendering/lights_and_shadows/directional_shadow/size

int	rendering/lights_and_shadows/directional_shadow/size.mobile
int	rendering/lights_and_shadows/directional_shadow/soft_shadow_filter_quality
int	rendering/lights_and_shadows/directional_shadow/soft_shadow_filter_quality.mobile
bool	rendering/lights_and_shadows/positional_shadow/atlas_16_bits
int	rendering/lights_and_shadows/positional_shadow/atlas_quadrant_0_subdiv
int	rendering/lights_and_shadows/positional_shadow/atlas_quadrant_1_subdiv
int	rendering/lights_and_shadows/positional_shadow/atlas_quadrant_2_subdiv
int	rendering/lights_and_shadows/positional_shadow/atlas_quadrant_3_subdiv
int	rendering/lights_and_shadows/positional_shadow/atlas_size
int	rendering/lights_and_shadows/positional_shadow/atlas_size.mobile
int	rendering/lights_and_shadows/positional_shadow/soft_shadow_filter_quality
int	rendering/lights_and_shadows/positional_shadow/soft_shadow_filter_quality.mobile
bool	rendering/lights_and_shadows/tighter_shadow_caster_culling
bool	rendering/lights_and_shadows/use_physical_light_units
float	rendering/limits/cluster_builder/max_clustered_elements
int	rendering/limits/global_shader_variables/buffer_size
int	rendering/limits/opengl/max_lights_per_object
int	rendering/limits/opengl/max_renderable_elements
int	rendering/limits/opengl/max_renderable_lights

int	rendering/limits/spatial_indexer/threaded_cull_minimum_instances
int	rendering/limits/spatial_indexer/update_iterations_per_frame
float	rendering/limits/time/time_rollover_secs
float	rendering/mesh_lod/lod_change/threshold_pixels
int	rendering/occlusion_culling/bvh_build_quality
bool	rendering/occlusion_culling/jitter_projection
int	rendering/occlusion_culling/occlusion_rays_per_thread
bool	rendering/occlusion_culling/use_occlusion_culling
int	rendering/reflections/reflection_atlas/reflection_count
int	rendering/reflections/reflection_atlas/reflection_size
int	rendering/reflections/reflection_atlas/reflection_size.mobile
bool	rendering/reflections/sky_reflections/fast_filter_high_quality
int	rendering/reflections/sky_reflections/ggx_samples
int	rendering/reflections/sky_reflections/ggx_samples.mobile
int	rendering/reflections/sky_reflections/roughness_layers
bool	rendering/reflections/sky_reflections/texture_array_reflections
bool	rendering/reflections/sky_reflections/texture_array_reflections.mobile
String	rendering/renderer/rendering_method
String	rendering/renderer/rendering_method.mobile

String	rendering/renderer/rendering_method.web
int	rendering/rendering_device/d3d12/agility_sdk_version
int	rendering/rendering_device/d3d12/max_misc_descriptors_per_frame
int	rendering/rendering_device/d3d12/max_resource_descriptors_per_frame
int	rendering/rendering_device/d3d12/max_sampler_descriptors_per_frame
String	rendering/rendering_device/driver
String	rendering/rendering_device/driver.android
String	rendering/rendering_device/driver.ios
String	rendering/rendering_device/driver.linuxbsd
String	rendering/rendering_device/driver.macos
String	rendering/rendering_device/driver.windows
bool	rendering/rendering_device/fallback_to_d3d12
bool	rendering/rendering_device/fallback_to_vulkan
bool	rendering/rendering_device/pipeline_cache/enable
float	rendering/rendering_device/pipeline_cache/save_chunk_size_mb
int	rendering/rendering_device/staging_buffer/block_size_kb
int	rendering/rendering_device/staging_buffer/max_size_mb
int	rendering/rendering_device/staging_buffer/texture_upload_region_size_px
int	rendering/rendering_device/vsync/frame_queue_size

int	rendering/rendering_device/vsync/swapchain_image_count
int	rendering/rendering_device/vulkan/max_descriptors_per_pool
float	rendering/scaling_3d/fsr_sharpness
int	rendering/scaling_3d/mode
float	rendering/scaling_3d/scale
bool	rendering/shader_compiler/shader_cache/compress
bool	rendering/shader_compiler/shader_cache/enabled
bool	rendering/shader_compiler/shader_cache/strip_debug
bool	rendering/shader_compiler/shader_cache/strip_debug.release
bool	rendering/shader_compiler/shader_cache/use_zstd_compression
bool	rendering/shading/overrides/force_lambert_over_burley
bool	rendering/shading/overrides/force_lambert_over_burley.mobile
bool	rendering/shading/overrides/force_vertex_shading
bool	rendering/shading/overrides/force_vertex_shading.mobile
int	rendering/textures/canvas_textures/default_texture_filter
int	rendering/textures/canvas_textures/default_texture_repeat
int	rendering/textures/decals/filter
int	rendering/textures/default_filters/anisotropic_filtering_level
float	rendering/textures/default_filters/texture_mipmap_bias

bool	rendering/textures/default_filters/use_nearest_mipmap_filter
int	rendering/textures/light_projectors/filter
bool	rendering/textures/lossless_compression/force_png
bool	rendering/textures/vram_compression/import_etc2_astc
bool	rendering/textures/vram_compression/import_s3tc_bptc
int	rendering/textures/webp_compression/compression_method
float	rendering/textures/webp_compression/lossless_compression_factor
bool	rendering/viewport/hdr_2d
bool	rendering/viewport/transparent_background
int	rendering/vrs/mode
String	rendering/vrs/texture
float	threading/worker_pool/low_priority_thread_ratio
int	threading/worker_pool/max_threads
String	xr/openxr/default_action_map
bool	xr/openxr/enabled
int	xr/openxr/environment_blend_mode
bool	xr/openxr/extensions/eye_gaze_interaction
bool	xr/openxr/extensions/hand_interaction_profile
bool	xr/openxr/extensions/hand_tracking

int	xr/openxr/form_factor
bool	xr/openxr/foveation_dynamic
int	xr/openxr/foveation_level
int	xr/openxr/reference_space
bool	xr/openxr/startup_alert
bool	xr/openxr/submit_depth_buffer
int	xr/openxr/view_configuration
bool	xr/shaders/enabled

Methods

void	add_property_info(hint: Dictionary)
void	clear(name: String)
Array[Dictionary]	get_global_class_list()
int	get_order(name: String) const
Variant	get_setting(name: String, default_value: Variant = null) const
Variant	get_setting_with_override(name: StringName) const
String	globalize_path(path: String) const
bool	has_setting(name: String) const
bool	load_resource_pack(pack: String, replace_files: bool = true, offset: int = 0)
String	localize_path(path: String) const

Error	save()
Error	save_custom(file: String)
void	set_as_basic(name: String, basic: bool)
void	set_as_internal(name: String, internal: bool)
void	set_initial_value(name: String, value: Variant)
void	set_order(name: String, position: int)
void	set_restart_if_changed(name: String, restart: bool)
void	set_setting(name: String, value: Variant)

Signals

settings_changed()

Emitted when any setting is changed, up to once per process frame.

Property Descriptions

bool **animation/warnings/check_angle_interpolation_type_conflicting** = `true`

If `true`, AnimationMixer prints the warning of interpolation being forced to choose the shortest rotation path due to multiple angle interpolation types being mixed in the AnimationMixer cache.

bool **animation/warnings/check_invalid_track_paths** = `true`

If `true`, AnimationMixer prints the warning of no matching object of the track path in the scene.

Color **application/boot_splash/bg_color** = `Color(0.14, 0.14, 0.14, 1)`

Background color for the boot splash.

bool **application/boot_splash/fullsize** = `true`

If `true`, scale the boot splash image to the full window size (preserving the aspect ratio) when the engine starts. If `false`, the engine will leave it at the default pixel size.

String **application/boot_splash/image** = `""` 🔗

Path to an image used as the boot splash. If left empty, the default Godot Engine splash will be displayed instead.

Note: Only effective if application/boot_splash/show_image is `true`.

Note: The only supported format is PNG. Using another image format will result in an error.

Note: The image will also show when opening the project in the editor. If you want to display the default splash image in the editor, add an empty override for `editor_hint` feature.

---

int **application/boot_splash/minimum_display_time** = `0` 🔗

Minimum boot splash display time (in milliseconds). It is not recommended to set too high values for this setting.

---

bool **application/boot_splash/show_image** = `true` 🔗

If `true`, displays the image specified in application/boot_splash/image when the engine starts. If `false`, only displays the plain color specified in application/boot_splash/bg_color.

---

bool **application/boot_splash/use_filter** = `true` 🔗

If `true`, applies linear filtering when scaling the image (recommended for high-resolution artwork). If `false`, uses nearest-neighbor interpolation (recommended for pixel art).

---

bool **application/config/auto_accept_quit** = `true` 🔗

If `true`, the application automatically accepts quitting requests.

String **application/config/custom_user_dir_name** = `""` 🔗

This user directory is used for storing persistent data (`user://` filesystem). If a custom directory name is defined, this name will be appended to the system-specific user data directory (same parent folder as the Godot configuration folder documented in OS.get_user_data_dir).

The application/config/use_custom_user_dir setting must be enabled for this to take effect.

Note: If application/config/custom_user_dir_name contains trailing periods, they will be stripped as folder names ending with a period are not allowed on Windows.

String **application/config/description** = `""` 🔗

The project's description, displayed as a tooltip in the Project Manager when hovering the project.

String **application/config/icon** = `""`

Icon used for the project, set when project loads. Exporters will also use this icon as a fallback if necessary.

String **application/config/macos_native_icon** = `""`

Icon set in `.icns` format used on macOS to set the game's icon. This is done automatically on start by calling DisplayServer.set_native_icon.

String **application/config/name** = `""`

The project's name. It is used both by the Project Manager and by exporters. The project name can be translated by translating its value in localization files. The window title will be set to match the project name automatically on startup.

Note: Changing this value will also change the user data folder's path if application/config/use_custom_user_dir is `false`. After renaming the project, you will no longer be able to access existing data in `user://` unless you rename the old folder to match the new project name. See Data paths in the documentation for more information.

Dictionary **application/config/name_localized** = `{}`

Translations of the project's name. This setting is used by OS tools to translate application name on Android, iOS and macOS.

String **application/config/project_settings_override** = `""`

Specifies a file to override project settings. For example: `user://custom_settings.cfg`. See "Overriding" in the **ProjectSettings** class description at the top for more information.

Note: Regardless of this setting's value, `res://override.cfg` will still be read to override the project settings.

bool **application/config/quit_on_go_back** = `true`

If `true`, the application quits automatically when navigating back (e.g. using the system "Back" button on Android).

bool **application/config/use_custom_user_dir** = `false`

If `true`, the project will save user data to its own user directory. If application/config/custom_user_dir_name is empty, `<OS user data directory>/<project name>` directory will be used. If `false`, the project will save user data to `<OS user data directory>/Godot/app_userdata/<project name>`.

See also File paths in Godot projects. This setting is only effective on desktop platforms.

bool **application/config/use_hidden_project_data_directory** = `true` 🔗

If `true`, the project will use a hidden directory (`.godot`) for storing project-specific data (metadata, shader cache, etc.).

If `false`, a non-hidden directory (`godot`) will be used instead.

Note: Restart the application after changing this setting.

Note: Changing this value can help on platforms or with third-party tools where hidden directory patterns are disallowed. Only modify this setting if you know that your environment requires it, as changing the default can impact compatibility with some external tools or plugins which expect the default `.godot` folder.

String **application/config/version** = `""` 🔗

The project's human-readable version identifier. This is used by exporters if the version identifier isn't overridden there. If application/config/version is an empty string and the version identifier isn't overridden in an exporter, the exporter will use `1.0.0` as a version identifier.

String **application/config/windows_native_icon** = `""` 🔗

Icon set in `.ico` format used on Windows to set the game's icon. This is done automatically on start by calling DisplayServer.set_native_icon.

---

bool **application/run/delta_smoothing** = `true` 🔗

Time samples for frame deltas are subject to random variation introduced by the platform, even when frames are displayed at regular intervals thanks to V-Sync. This can lead to jitter. Delta smoothing can often give a better result by filtering the input deltas to correct for minor fluctuations from the refresh rate.

Note: Delta smoothing is only attempted when display/window/vsync/vsync_mode is set to `enabled`, as it does not work well without V-Sync.

It may take several seconds at a stable frame rate before the smoothing is initially activated. It will only be active on machines where performance is adequate to render frames at the refresh rate.

---

bool **application/run/disable_stderr** = `false` 🔗

If `true`, disables printing to standard error. If `true`, this also hides error and warning messages printed by @GlobalScope.push_error and @GlobalScope.push_warning. See also application/run/disable_stdout.

Changes to this setting will only be applied upon restarting the application.

---

bool **application/run/disable_stdout** = `false` 🔗

If `true`, disables printing to standard output. This is equivalent to starting the editor or project with the `--quiet` command line argument. See also application/run/disable_stderr.

Changes to this setting will only be applied upon restarting the application.

bool **application/run/enable_alt_space_menu** = `false`

If `true`, allows the `Alt + Space` keys to display the window menu. This menu allows the user to perform various window management operations such as moving, resizing, or minimizing the window.

Note: When the menu is displayed, project execution will pause until the menu is *fully* closed due to Windows behavior. Consider this when enabling this setting in a networked multiplayer game. The menu is only considered fully closed when an option is selected, when the user clicks outside, or when `Escape` is pressed after bringing up the window menu *and* another key is pressed afterwards.

Note: This setting is implemented only on Windows.

bool **application/run/flush_stdout_on_print** = `false`

If `true`, flushes the standard output stream every time a line is printed. This affects both terminal logging and file logging.

When running a project, this setting must be enabled if you want logs to be collected by service managers such as systemd/journalctl. This setting is disabled by default on release builds, since flushing on every printed line will negatively affect performance if lots of lines are printed in a rapid succession. Also, if this setting is enabled, logged files will still be written successfully if the application crashes or is otherwise killed by the user (without being closed "normally").

Note: Regardless of this setting, the standard error stream (`stderr`) is always flushed when a line is printed to it.

Changes to this setting will only be applied upon restarting the application.

bool **application/run/flush_stdout_on_print.debug** = `true`

Debug build override for application/run/flush_stdout_on_print, as performance is less important during debugging.

Changes to this setting will only be applied upon restarting the application.

int **application/run/frame_delay_msec** = `0`

Forces a *constant* delay between frames in the main loop (in milliseconds). In most situations, application/run/max_fps should be preferred as an FPS limiter as it's more precise.

This setting can be overridden using the `--frame-delay <ms;>` command line argument.

bool **application/run/low_processor_mode** = `false`

If `true`, enables low-processor usage mode. When enabled, the engine takes longer to redraw, but only redraws the screen if necessary. This may lower power consumption, and is intended for editors or mobile applications. For most games, because the screen needs to be redrawn every frame, it is recommended to keep this setting disabled.

int **application/run/low_processor_mode_sleep_usec** = `6900`

Amount of sleeping between frames when the low-processor usage mode is enabled (in microseconds). Higher values will result in lower CPU usage.

String **application/run/main_loop_type** = `"SceneTree"`

The name of the type implementing the engine's main loop.

String **application/run/main_scene** = `""`

Path to the main scene file that will be loaded when the project runs.

---

int **application/run/max_fps** = `0`

Maximum number of frames per second allowed. A value of `0` means "no limit". The actual number of frames per second may still be below this value if the CPU or GPU cannot keep up with the project logic and rendering.

Limiting the FPS can be useful to reduce system power consumption, which reduces heat and noise emissions (and improves battery life on mobile devices).

If display/window/vsync/vsync_mode is set to `Enabled` or `Adaptive`, it takes precedence and the forced FPS number cannot exceed the monitor's refresh rate.

If display/window/vsync/vsync_mode is `Enabled`, on monitors with variable refresh rate enabled (G-Sync/FreeSync), using an FPS limit a few frames lower than the monitor's refresh rate will reduce input lag while avoiding tearing [https://blurbusters.com/howto-low-lag-vsync-on/].

If display/window/vsync/vsync_mode is `Disabled`, limiting the FPS to a high value that can be consistently reached on the system can reduce input lag compared to an uncapped framerate. Since this works by ensuring the GPU load is lower than 100%, this latency reduction is only effective in GPU-bottlenecked scenarios, not CPU-bottlenecked scenarios.

See also physics/common/physics_ticks_per_second.

This setting can be overridden using the `--max-fps <fps>` command line argument (including with a value of `0` for unlimited framerate).

Note: This property is only read when the project starts. To change the rendering FPS cap at runtime, set Engine.max_fps instead.

---

bool **application/run/print_header** = `true`

If `true`, the engine header is printed in the console on startup. This header describes the current version of the engine, as well as the renderer being used. This behavior can also be disabled on the command line with the `--no-header` option.

---

float **audio/buses/channel_disable_threshold_db** = `-60.0`

Audio buses will disable automatically when sound goes below a given dB threshold for a given time. This saves CPU as effects assigned to that bus will no longer do any processing.

float **audio/buses/channel_disable_time** = `2.0`

Audio buses will disable automatically when sound goes below a given dB threshold for a given time. This saves CPU as effects assigned to that bus will no longer do any processing.

String **audio/buses/default_bus_layout** = `"res://default_bus_layout.tres"`

Default AudioBusLayout resource file to use in the project, unless overridden by the scene.

String **audio/driver/driver**

Specifies the audio driver to use. This setting is platform-dependent as each platform supports different audio drivers. If left empty, the default audio driver will be used.

The `Dummy` audio driver disables all audio playback and recording, which is useful for non-game applications as it reduces CPU usage. It also prevents the engine from appearing as an application playing audio in the OS' audio mixer.

Note: The driver in use can be overridden at runtime via the `--audio-driver` command line argument.

bool **audio/driver/enable_input** = `false`

If `true`, microphone input will be allowed. This requires appropriate permissions to be set when exporting to Android or iOS.

Note: If the operating system blocks access to audio input devices (due to the user's privacy settings), audio capture will only return silence. On Windows 10 and later, make sure that apps are allowed to access the microphone in the OS' privacy settings.

int **audio/driver/mix_rate** = `44100`

The mixing rate used for audio (in Hz). In general, it's better to not touch this and leave it to the host operating system.

int **audio/driver/mix_rate.web** = `0`

Safer override for audio/driver/mix_rate in the Web platform. Here `0` means "let the browser choose" (since some browsers do not like forcing the mix rate).

int **audio/driver/output_latency** = `15`

Specifies the preferred output latency in milliseconds for audio. Lower values will result in lower audio latency at the cost of increased CPU usage. Low values may result in audible crackling on slower hardware.

Audio output latency may be constrained by the host operating system and audio hardware drivers. If the host can not provide the specified audio output latency then Godot will attempt to use the nearest latency allowed by the host. As such you should always use AudioServer.get_output_latency to determine the actual audio output latency.

Audio output latency can be overridden using the `--audio-output-latency <ms>` command line argument.

Note: This setting is ignored on Android, and on all versions of Windows prior to Windows 10.

---

int **audio/driver/output_latency.web** = `50`

Safer override for audio/driver/output_latency in the Web platform, to avoid audio issues especially on mobile devices.

---

float **audio/general/2d_panning_strength** = `0.5`

The base strength of the panning effect for all AudioStreamPlayer2D nodes. The panning strength can be further scaled on each Node using AudioStreamPlayer2D.panning_strength. A value of `0.0` disables stereo panning entirely, leaving only volume attenuation in place. A value of `1.0` completely mutes one of the channels if the sound is located exactly to the left (or right) of the listener.

The default value of `0.5` is tuned for headphones. When using speakers, you may find lower values to sound better as speakers have a lower stereo separation compared to headphones.

---

float **audio/general/3d_panning_strength** = `0.5`

The base strength of the panning effect for all AudioStreamPlayer3D nodes. The panning strength can be further scaled on each Node using AudioStreamPlayer3D.panning_strength. A value of `0.0` disables stereo panning entirely, leaving only volume attenuation in place. A value of `1.0` completely mutes one of the channels if the sound is located exactly to the left (or right) of the listener.

The default value of `0.5` is tuned for headphones. When using speakers, you may find lower values to sound better as speakers have a lower stereo separation compared to headphones.

---

int **audio/general/default_playback_type** = `0`

Experimental: This property may be changed or removed in future versions.

Specifies the default playback type of the platform.

The default value is set to **Stream**, as most platforms have no issues mixing streams.

---

int **audio/general/default_playback_type.web** = `1`

Experimental: This property may be changed or removed in future versions.

Specifies the default playback type of the Web platform.

The default value is set to **Sample** as the Web platform is not suited to mix audio streams outside of the Web Audio API, especially when exporting a single-threaded game. **Sample** allows for lower latency on the web platform at the cost of flexibility (AudioEffects are not supported).

Warning: Forcing **Stream** on the Web platform may cause high audio latency and crackling, especially when exporting a multi-threaded game.

---

bool **audio/general/ios/mix_with_others** = `false`

Sets the mixWithOthers [https://developer.apple.com/documentation/avfaudio/avaudiosession/categoryoptions/1616611-mixwithothers] option for the AVAudioSession on iOS. This will override the mix behavior, if the category is set to `Play and Record`, `Playback`, or `Multi Route`.

`Ambient` always has this set per default.

---

int **audio/general/ios/session_category** = `0`

Sets the AVAudioSessionCategory [https://developer.apple.com/documentation/avfaudio/avaudiosessioncategory] on iOS. Use the `Playback` category to get sound output, even if the phone is in silent mode.

---

bool **audio/general/text_to_speech** = `false`

If `true`, text-to-speech support is enabled, see DisplayServer.tts_get_voices and DisplayServer.tts_speak.

Note: Enabling TTS can cause addition idle CPU usage and interfere with the sleep mode, so consider disabling it if TTS is not used.

---

int **audio/video/video_delay_compensation_ms** = `0`

Setting to hardcode audio delay when playing video. Best to leave this unchanged unless you know what you are doing.

---

bool **collada/use_ambient** = `false`

If `true`, ambient lights will be imported from COLLADA models as DirectionalLight3D. If `false`, ambient lights will be ignored.

---

int **compression/formats/gzip/compression_level** = `-1`

The default compression level for gzip. Affects compressed scenes and resources. Higher levels result in smaller files at the cost of compression speed. Decompression speed is mostly unaffected by the compression level. `-1` uses the default gzip compression level, which is identical to `6` but could change in the future due to underlying zlib updates.

---

int **compression/formats/zlib/compression_level** = `-1`

The default compression level for Zlib. Affects compressed scenes and resources. Higher levels result in smaller files at the cost of compression speed. Decompression speed is mostly unaffected by the compression level. `-1` uses the default gzip compression level, which is identical to `6` but could change in the future due to underlying zlib updates.

---

int **compression/formats/zstd/compression_level** = `3`

The default compression level for Zstandard. Affects compressed scenes and resources. Higher levels result in smaller files at the cost of compression speed. Decompression speed is mostly unaffected by the compression level.

---

bool **compression/formats/zstd/long_distance_matching** = `false`

Enables long-distance matching [https://github.com/facebook/zstd/releases/tag/v1.3.2] in Zstandard.

---

int **compression/formats/zstd/window_log_size** = `27`

Largest size limit (in power of 2) allowed when compressing using long-distance matching with Zstandard. Higher values can result in better compression, but will require more memory when compressing and decompressing.

Color **debug/canvas_items/debug_redraw_color** = `Color(1, 0.2, 0.2, 0.5)`

If canvas item redraw debugging is active, this color will be flashed on canvas items when they redraw.

---

float **debug/canvas_items/debug_redraw_time** = `1.0`

If canvas item redraw debugging is active, this will be the time the flash will last each time they redraw.

---

bool **debug/file_logging/enable_file_logging** = `false`

If `true`, logs all output and error messages to files. See also debug/file_logging/log_path, debug/file_logging/max_log_files, and application/run/flush_stdout_on_print.

---

bool **debug/file_logging/enable_file_logging.pc** = `true`

Desktop override for debug/file_logging/enable_file_logging, as log files are not readily accessible on mobile/Web platforms.

String **debug/file_logging/log_path** = `"user://logs/godot.log"`

Path at which to store log files for the project. Using a path under `user://` is recommended.

This can be specified manually on the command line using the `--log-file <file>` command line argument. If this command line argument is specified, log rotation is automatically disabled (see debug/file_logging/max_log_files).

---

int **debug/file_logging/max_log_files** = `5`

Specifies the maximum number of log files allowed (used for rotation). Set to `1` to disable log file rotation.

If the `--log-file <file>` command line argument is used, log rotation is always disabled.

---

int **debug/gdscript/warnings/assert_always_false** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when an `assert` call always evaluates to false.

---

int **debug/gdscript/warnings/assert_always_true** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when an `assert` call always evaluates to true.

---

int **debug/gdscript/warnings/confusable_capture_reassignment** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when a local variable captured by a lambda is reassigned, since this does not modify the outer local variable.

---

int **debug/gdscript/warnings/confusable_identifier** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when an identifier contains characters that can be confused with something else, like when mixing different alphabets.

---

int **debug/gdscript/warnings/confusable_local_declaration** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when an identifier declared in the nested block has the same name as an identifier declared below in the parent block.

---

int **debug/gdscript/warnings/confusable_local_usage** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when an identifier that will be shadowed below in the block is used.

---

int **debug/gdscript/warnings/constant_used_as_function** = `1`

Deprecated: This warning is never produced. Instead, an error is generated if the expression type is known at compile time.

When set to `warn` or `error`, produces a warning or an error respectively when a constant is used as a function.

---

int **debug/gdscript/warnings/deprecated_keyword** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when deprecated keywords are used.

Note: There are currently no deprecated keywords, so this warning is never produced.

---

int **debug/gdscript/warnings/empty_file** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when an empty file is parsed.

---

bool **debug/gdscript/warnings/enable** = `true` 🔗

If `true`, enables specific GDScript warnings (see `debug/gdscript/warnings/*` settings). If `false`, disables all GDScript warnings.

---

int **debug/gdscript/warnings/enum_variable_without_default** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when a variable has an enum type but no explicit default value, but only if the enum does not contain `0` as a valid value.

---

bool **debug/gdscript/warnings/exclude_addons** = `true` 🔗

If `true`, scripts in the `res://addons` folder will not generate warnings.

---

int **debug/gdscript/warnings/function_used_as_property** = `1` 🔗

Deprecated: This warning is never produced. When a function is used as a property, a Callable is returned.

When set to `warn` or `error`, produces a warning or an error respectively when using a function as if it is a property.

---

int **debug/gdscript/warnings/get_node_default_without_onready** = `2` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when Node.get_node (or the shorthand `$`) is used as default value of a class variable without the `@onready` annotation.

---

int **debug/gdscript/warnings/incompatible_ternary** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when a ternary operator may emit values with incompatible types.

---

int **debug/gdscript/warnings/inference_on_variant** = `2` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when a static inferred type uses a Variant as initial value, which makes the static type to also be Variant.

---

int **debug/gdscript/warnings/inferred_declaration** = `0` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when a variable, constant, or parameter has an implicitly inferred static type.

Note: This warning is recommended *in addition* to debug/gdscript/warnings/untyped_declaration if you want to always specify the type explicitly. Having `INFERRED_DECLARATION` warning level higher than `UNTYPED_DECLARATION` warning level makes little sense and is not recommended.

---

int **debug/gdscript/warnings/int_as_enum_without_cast** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when trying to use an integer as an enum without an explicit cast.

---

int **debug/gdscript/warnings/int_as_enum_without_match** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when trying to use an integer as an enum when there is no matching enum member for that numeric value.

---

int **debug/gdscript/warnings/integer_division** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when dividing an integer by another integer (the decimal part will be discarded).

---

int **debug/gdscript/warnings/narrowing_conversion** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when passing a floating-point value to a function that expects an integer (it will be converted and lose precision).

---

int **debug/gdscript/warnings/native_method_override** = `2` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when a method in the script overrides a native method, because it may not behave as expected.

---

int **debug/gdscript/warnings/onready_with_export** = `2` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when the `@onready` annotation is used together with the `@export` annotation, since it may not behave as expected.

---

int **debug/gdscript/warnings/property_used_as_function** = `1` 🔗

Deprecated: This warning is never produced. Instead, an error is generated if the expression type is known at compile time.

When set to `warn` or `error`, produces a warning or an error respectively when using a property as if it is a function.

---

int **debug/gdscript/warnings/redundant_await** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when a function that is not a coroutine is called with await.

---

int **debug/gdscript/warnings/redundant_static_unload** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when the `@static_unload` annotation is used in a script without any static variables.

---

bool **debug/gdscript/warnings/renamed_in_godot_4_hint** = `1`

When enabled, using a property, enum, or function that was renamed since Godot 3 will produce a hint if an error occurs.

---

int **debug/gdscript/warnings/return_value_discarded** = `0`

When set to `warn` or `error`, produces a warning or an error respectively when calling a function without using its return value (by assigning it to a variable or using it as a function argument). These return values are sometimes used to indicate possible errors using the Error enum.

---

int **debug/gdscript/warnings/shadowed_global_identifier** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when defining a local or member variable, signal, or enum that would have the same name as a built-in function or global class name, thus shadowing it.

---

int **debug/gdscript/warnings/shadowed_variable** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when defining a local or member variable that would shadow a member variable that the class defines.

---

int **debug/gdscript/warnings/shadowed_variable_base_class** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when defining a local or subclass member variable that would shadow a variable that is inherited from a parent class.

---

int **debug/gdscript/warnings/standalone_expression** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when calling an expression that may have no effect on the surrounding code, such as writing `2 + 2` as a statement.

---

int **debug/gdscript/warnings/standalone_ternary** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when calling a ternary expression that may have no effect on the surrounding code, such as writing `42 if active else 0` as a statement.

---

int **debug/gdscript/warnings/static_called_on_instance** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when calling a static method from an instance of a class instead of from the class directly.

---

int **debug/gdscript/warnings/unassigned_variable** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when using a variable that wasn't previously assigned.

---

int **debug/gdscript/warnings/unassigned_variable_op_assign** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when assigning a variable using an assignment operator like `+=` if the variable wasn't previously assigned.

---

int **debug/gdscript/warnings/unreachable_code** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when unreachable code is detected (such as after a `return` statement that will always be executed).

---

int **debug/gdscript/warnings/unreachable_pattern** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when an unreachable `match` pattern is detected.

---

int **debug/gdscript/warnings/unsafe_call_argument** = `0` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when using an expression whose type may not be compatible with the function parameter expected.

---

int **debug/gdscript/warnings/unsafe_cast** = `0` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when a Variant value is cast to a non-Variant.

---

int **debug/gdscript/warnings/unsafe_method_access** = `0` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when calling a method whose presence is not guaranteed at compile-time in the class.

---

int **debug/gdscript/warnings/unsafe_property_access** = `0` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when accessing a property whose presence is not guaranteed at compile-time in the class.

---

int **debug/gdscript/warnings/unsafe_void_return** = `1` 🔗

When set to `warn` or `error`, produces a warning or an error respectively when returning a call from a `void` function when such call cannot be guaranteed to be also `void`.

---

int **debug/gdscript/warnings/untyped_declaration** = `0`

When set to `warn` or `error`, produces a warning or an error respectively when a variable or parameter has no static type, or if a function has no static return type.

Note: This warning is recommended together with EditorSettings.text_editor/completion/add_type_hints to help achieve type safety.

---

int **debug/gdscript/warnings/unused_local_constant** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when a local constant is never used.

---

int **debug/gdscript/warnings/unused_parameter** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when a function parameter is never used.

---

int **debug/gdscript/warnings/unused_private_class_variable** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when a private member variable is never used.

---

int **debug/gdscript/warnings/unused_signal** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when a signal is declared but never explicitly used in the class.

---

int **debug/gdscript/warnings/unused_variable** = `1`

When set to `warn` or `error`, produces a warning or an error respectively when a local variable is unused.

String **debug/settings/crash_handler/message** = `"Please include this when reporting the bug to the project developer."`

Message to be displayed before the backtrace when the engine crashes. By default, this message is only used in exported projects due to the editor-only override applied to this setting.

String **debug/settings/crash_handler/message.editor** = `"Please include this when reporting the bug on: https://github.com/godotengine/godot/issues"`

Editor-only override for debug/settings/crash_handler/message. Does not affect exported projects in debug or release mode.

---

int **debug/settings/gdscript/max_call_stack** = `1024`

Maximum call stack allowed for debugging GDScript.

---

int **debug/settings/profiler/max_functions** = `16384`

Maximum number of functions per frame allowed when profiling.

---

int **debug/settings/profiler/max_timestamp_query_elements** = `256`

Maximum number of timestamp query elements allowed per frame for visual profiling.

---

bool **debug/settings/stdout/print_fps** = `false`

Print frames per second to standard output every second.

---

bool **debug/settings/stdout/print_gpu_profile** = `false`

Print GPU profile information to standard output every second. This includes how long each frame takes the GPU to render on average, broken down into different steps of the render pipeline, such as CanvasItems, shadows, glow, etc.

---

bool **debug/settings/stdout/verbose_stdout** = `false`

Print more information to standard output when running. It displays information such as memory leaks, which scenes and resources are being loaded, etc. This can also be enabled using the `--verbose` or `-v` command line argument, even on an exported project. See also OS.is_stdout_verbose and @GlobalScope.print_verbose.

---

bool **debug/shader_language/warnings/device_limit_exceeded** = `true`

When set to `true`, produces a warning when the shader exceeds certain device limits. Currently, the only device limit checked is the limit on uniform buffer size. More device limits will be added in the future.

---

bool **debug/shader_language/warnings/enable** = `true`

If `true`, enables specific shader warnings (see `debug/shader_language/warnings/*` settings). If `false`, disables all shader warnings.

---

bool **debug/shader_language/warnings/float_comparison** = `true`

When set to `true`, produces a warning when two floating-point numbers are compared directly with the `==` operator or the `!=` operator.

---

bool **debug/shader_language/warnings/formatting_error** = `true`

When set to `true`, produces a warning upon encountering certain formatting errors. Currently this only checks for empty statements. More formatting errors may be added over time.

bool **debug/shader_language/warnings/magic_position_write** = `true`

When set to `true`, produces a warning when the shader contains `POSITION = vec4(vertex,` as this was very common code written in Godot 4.2 and earlier that was paired with a QuadMesh to produce a full screen post processes pass. With the switch to reversed z in 4.3, this trick no longer works, as it implicitly relied on the `VERTEX.z` being 0.

bool **debug/shader_language/warnings/treat_warnings_as_errors** = `false`

When set to `true`, warnings are treated as errors.

bool **debug/shader_language/warnings/unused_constant** = `true`

When set to `true`, produces a warning when a constant is never used.

bool **debug/shader_language/warnings/unused_function** = `true`

When set to `true`, produces a warning when a function is never used.

bool **debug/shader_language/warnings/unused_local_variable** = `true`

When set to `true`, produces a warning when a local variable is never used.

bool **debug/shader_language/warnings/unused_struct** = `true`

When set to `true`, produces a warning when a struct is never used.

bool **debug/shader_language/warnings/unused_uniform** = `true`

When set to `true`, produces a warning when a uniform is never used.

bool **debug/shader_language/warnings/unused_varying** = `true`

When set to `true`, produces a warning when a varying is never used.

Color **debug/shapes/avoidance/agents_radius_color** = `Color(1, 1, 0, 0.25)`

Color of the avoidance agents radius, visible when "Visible Avoidance" is enabled in the Debug menu.

bool **debug/shapes/avoidance/enable_agents_radius** = `true`

If enabled, displays avoidance agents radius when "Visible Avoidance" is enabled in the Debug menu.

---

bool **debug/shapes/avoidance/enable_obstacles_radius** = `true`

If enabled, displays avoidance obstacles radius when "Visible Avoidance" is enabled in the Debug menu.

---

bool **debug/shapes/avoidance/enable_obstacles_static** = `true`

If enabled, displays static avoidance obstacles when "Visible Avoidance" is enabled in the Debug menu.

Color **debug/shapes/avoidance/obstacles_radius_color** = `Color(1, 0.5, 0, 0.25)`

Color of the avoidance obstacles radius, visible when "Visible Avoidance" is enabled in the Debug menu.

Color **debug/shapes/avoidance/obstacles_static_edge_pushin_color** = `Color(1, 0, 0, 1)`

Color of the static avoidance obstacles edges when their vertices are winded in order to push agents in, visible when "Visible Avoidance" is enabled in the Debug menu.

Color **debug/shapes/avoidance/obstacles_static_edge_pushout_color** = `Color(1, 1, 0, 1)`

Color of the static avoidance obstacles edges when their vertices are winded in order to push agents out, visible when "Visible Avoidance" is enabled in the Debug menu.

Color **debug/shapes/avoidance/obstacles_static_face_pushin_color** = `Color(1, 0, 0, 0)`

Color of the static avoidance obstacles faces when their vertices are winded in order to push agents in, visible when "Visible Avoidance" is enabled in the Debug menu.

Color **debug/shapes/avoidance/obstacles_static_face_pushout_color** = `Color(1, 1, 0, 0.5)`

Color of the static avoidance obstacles faces when their vertices are winded in order to push agents out, visible when "Visible Avoidance" is enabled in the Debug menu.

Color **debug/shapes/collision/contact_color** = `Color(1, 0.2, 0.1, 0.8)`

Color of the contact points between collision shapes, visible when "Visible Collision Shapes" is enabled in the Debug menu.

---

bool **debug/shapes/collision/draw_2d_outlines** = `true`

Sets whether 2D physics will display collision outlines in game when "Visible Collision Shapes" is enabled in the Debug menu.

---

int **debug/shapes/collision/max_contacts_displayed** = `10000`

Maximum number of contact points between collision shapes to display when "Visible Collision Shapes" is enabled in the Debug menu.

Color **debug/shapes/collision/shape_color** = `Color(0, 0.6, 0.7, 0.42)`

Color of the collision shapes, visible when "Visible Collision Shapes" is enabled in the Debug menu.

Color **debug/shapes/navigation/agent_path_color** = `Color(1, 0, 0, 1)`

Color to display enabled navigation agent paths when an agent has debug enabled.

---

float **debug/shapes/navigation/agent_path_point_size** = `4.0`

Rasterized size (pixel) used to render navigation agent path points when an agent has debug enabled.

Color **debug/shapes/navigation/edge_connection_color** = `Color(1, 0, 1, 1)`

Color to display edge connections between navigation regions, visible when "Visible Navigation" is enabled in the Debug menu.

---

bool **debug/shapes/navigation/enable_agent_paths** = `true`

If enabled, displays navigation agent paths when an agent has debug enabled.

---

bool **debug/shapes/navigation/enable_agent_paths_xray** = `true`

If enabled, displays navigation agent paths through geometry when an agent has debug enabled.

---

bool **debug/shapes/navigation/enable_edge_connections** = `true`

If enabled, displays edge connections between navigation regions when "Visible Navigation" is enabled in the Debug menu.

---

bool **debug/shapes/navigation/enable_edge_connections_xray** = `true`

If enabled, displays edge connections between navigation regions through geometry when "Visible Navigation" is enabled in the Debug menu.

---

bool **debug/shapes/navigation/enable_edge_lines** = `true`

If enabled, displays navigation mesh polygon edges when "Visible Navigation" is enabled in the Debug menu.

---

bool **debug/shapes/navigation/enable_edge_lines_xray** = `true`

If enabled, displays navigation mesh polygon edges through geometry when "Visible Navigation" is enabled in the Debug menu.

bool **debug/shapes/navigation/enable_geometry_face_random_color** = `true`

If enabled, colorizes each navigation mesh polygon face with a random color when "Visible Navigation" is enabled in the Debug menu.

bool **debug/shapes/navigation/enable_link_connections** = `true`

If enabled, displays navigation link connections when "Visible Navigation" is enabled in the Debug menu.

bool **debug/shapes/navigation/enable_link_connections_xray** = `true`

If enabled, displays navigation link connections through geometry when "Visible Navigation" is enabled in the Debug menu.

Color **debug/shapes/navigation/geometry_edge_color** = `Color(0.5, 1, 1, 1)`

Color to display enabled navigation mesh polygon edges, visible when "Visible Navigation" is enabled in the Debug menu.

Color **debug/shapes/navigation/geometry_edge_disabled_color** = `Color(0.5, 0.5, 0.5, 1)`

Color to display disabled navigation mesh polygon edges, visible when "Visible Navigation" is enabled in the Debug menu.

Color **debug/shapes/navigation/geometry_face_color** = `Color(0.5, 1, 1, 0.4)`

Color to display enabled navigation mesh polygon faces, visible when "Visible Navigation" is enabled in the Debug menu.

Color **debug/shapes/navigation/geometry_face_disabled_color** = `Color(0.5, 0.5, 0.5, 0.4)`

Color to display disabled navigation mesh polygon faces, visible when "Visible Navigation" is enabled in the Debug menu.

Color **debug/shapes/navigation/link_connection_color** = `Color(1, 0.5, 1, 1)`

Color to use to display navigation link connections, visible when "Visible Navigation" is enabled in the Debug menu.

Color **debug/shapes/navigation/link_connection_disabled_color** = `Color(0.5, 0.5, 0.5, 1)` 🔗

Color to use to display disabled navigation link connections, visible when "Visible Navigation" is enabled in the Debug menu.

Color **debug/shapes/paths/geometry_color** = `Color(0.1, 1, 0.7, 0.4)` 🔗

Color of the curve path geometry, visible when "Visible Paths" is enabled in the Debug menu.

---

float **debug/shapes/paths/geometry_width** = `2.0` 🔗

Line width of the curve path geometry, visible when "Visible Paths" is enabled in the Debug menu.

String **display/display_server/driver** 🔗

Sets the driver to be used by the display server. This property can not be edited directly, instead, set the driver using the platform-specific overrides.

String **display/display_server/driver.android** 🔗

Android override for display/display_server/driver.

String **display/display_server/driver.ios** 🔗

iOS override for display/display_server/driver.

String **display/display_server/driver.linuxbsd** 🔗

LinuxBSD override for display/display_server/driver.

String **display/display_server/driver.macos** 🔗

MacOS override for display/display_server/driver.

String **display/display_server/driver.windows** 🔗

Windows override for display/display_server/driver.

String **display/mouse_cursor/custom_image** = `""` 🔗

Custom image for the mouse cursor (limited to 256×256).

Vector2 **display/mouse_cursor/custom_image_hotspot** = `Vector2(0, 0)` 🔗

Hotspot for the custom mouse cursor image.

Vector2 **display/mouse_cursor/tooltip_position_offset** = `Vector2(10, 10)`

Position offset for tooltips, relative to the mouse cursor's hotspot.

bool **display/window/dpi/allow_hidpi** = `true`

If `true`, allows HiDPI display on Windows, macOS, Android, iOS and Web. If `false`, the platform's low-DPI fallback will be used on HiDPI displays, which causes the window to be displayed in a blurry or pixelated manner (and can cause various window management bugs). Therefore, it is recommended to make your project scale to multiple resolutions instead of disabling this setting.

Note: This setting has no effect on Linux as DPI-awareness fallbacks are not supported there.

bool **display/window/energy_saving/keep_screen_on** = `true`

If `true`, keeps the screen on (even in case of inactivity), so the screensaver does not take over. Works on desktop and mobile platforms.

bool **display/window/energy_saving/keep_screen_on.editor_hint** = `false`

Editor-only override for display/window/energy_saving/keep_screen_on. Does not affect running project.

int **display/window/handheld/orientation** = `0`

The default screen orientation to use on mobile devices. See ScreenOrientation for possible values.

Note: When set to a portrait orientation, this project setting does not flip the project resolution's width and height automatically. Instead, you have to set display/window/size/viewport_width and display/window/size/viewport_height accordingly.

bool **display/window/ios/allow_high_refresh_rate** = `true`

If `true`, iOS devices that support high refresh rate/"ProMotion" will be allowed to render at up to 120 frames per second.

bool **display/window/ios/hide_home_indicator** = `true`

If `true`, the home indicator is hidden automatically. This only affects iOS devices without a physical home button.

bool **display/window/ios/hide_status_bar** = `true`

If `true`, the status bar is hidden while the app is running.

bool **display/window/ios/suppress_ui_gesture** = `true`

If `true`, it will require two swipes to access iOS UI that uses gestures.

Note: This setting has no effect on the home indicator if `hide_home_indicator` is `true`.

---

bool **display/window/per_pixel_transparency/allowed** = `false`

If `true`, allows per-pixel transparency for the window background. This affects performance, so leave it on `false` unless you need it. See also display/window/size/transparent and rendering/viewport/transparent_background.

---

bool **display/window/size/always_on_top** = `false`

Forces the main window to be always on top.

Note: This setting is ignored on iOS, Android, and Web.

---

bool **display/window/size/borderless** = `false`

Forces the main window to be borderless.

Note: This setting is ignored on iOS, Android, and Web.

---

bool **display/window/size/extend_to_title** = `false`

Main window content is expanded to the full size of the window. Unlike a borderless window, the frame is left intact and can be used to resize the window, and the title bar is transparent, but has minimize/maximize/close buttons.

Note: This setting is implemented only on macOS.

Vector2i **display/window/size/initial_position** = `Vector2i(0, 0)`

Main window initial position (in virtual desktop coordinates), this setting is used only if display/window/size/initial_position_type is set to "Absolute" (`0`).

Note: This setting only affects the exported project, or when the project is run from the command line. In the editor, the value of EditorSettings.run/window_placement/rect_custom_position is used instead.

---

int **display/window/size/initial_position_type** = `1`

Main window initial position.

`0` - "Absolute", display/window/size/initial_position is used to set window position.

`1` - "Primary Screen Center".

`2` - "Other Screen Center", display/window/size/initial_screen is used to set the screen.

Note: This setting only affects the exported project, or when the project is run from the command line. In the editor, the value of EditorSettings.run/window_placement/rect is used instead.

---

int **display/window/size/initial_screen** = `0`

Main window initial screen, this setting is used only if display/window/size/initial_position_type is set to "Other Screen Center" (`2`).

Note: This setting only affects the exported project, or when the project is run from the command line. In the editor, the value of EditorSettings.run/window_placement/screen is used instead.

---

int **display/window/size/mode** = `0`

Main window mode. See WindowMode for possible values and how each mode behaves.

---

bool **display/window/size/no_focus** = `false`

Main window can't be focused. No-focus window will ignore all input, except mouse clicks.

---

bool **display/window/size/resizable** = `true`

If `true`, allows the window to be resizable by default.

Note: This property is only read when the project starts. To change whether the window is resizable at runtime, set Window.unresizable instead on the root Window, which can be retrieved using `get_viewport().get_window()`. Window.unresizable takes the opposite value of this setting.

Note: Certain window managers can be configured to ignore the non-resizable status of a window. Do not rely on this setting as a guarantee that the window will *never* be resizable.

Note: This setting is ignored on iOS.

---

bool **display/window/size/transparent** = `false`

If `true`, enables a window manager hint that the main window background *can* be transparent. This does not make the background actually transparent. For the background to be transparent, the root viewport must also be made transparent by enabling rendering/viewport/transparent_background.

Note: To use a transparent splash screen, set application/boot_splash/bg_color to `Color(0, 0, 0, 0)`.

Note: This setting has no effect if display/window/per_pixel_transparency/allowed is set to `false`.

---

int **display/window/size/viewport_height** = `648`

Sets the game's main viewport height. On desktop platforms, this is also the initial window height, represented by an indigo-colored rectangle in the 2D editor. Stretch mode settings also use this as a reference when using the `canvas_items` or `viewport` stretch modes. See also display/window/size/viewport_width, display/window/size/window_width_override and display/window/size/window_height_override.

---

int **display/window/size/viewport_width** = `1152`

Sets the game's main viewport width. On desktop platforms, this is also the initial window width, represented by an indigo-colored rectangle in the 2D editor. Stretch mode settings also use this as a reference when using the `canvas_items` or `viewport` stretch modes. See also display/window/size/viewport_height, display/window/size/window_width_override and display/window/size/window_height_override.

---

int **display/window/size/window_height_override** = `0`

On desktop platforms, overrides the game's initial window height. See also display/window/size/window_width_override, display/window/size/viewport_width and display/window/size/viewport_height.

Note: By default, or when set to `0`, the initial window height is the display/window/size/viewport_height. This setting is ignored on iOS, Android, and Web.

---

int **display/window/size/window_width_override** = `0`

On desktop platforms, overrides the game's initial window width. See also display/window/size/window_height_override, display/window/size/viewport_width and display/window/size/viewport_height.

Note: By default, or when set to `0`, the initial window width is the display/window/size/viewport_width. This setting is ignored on iOS, Android, and Web.

String **display/window/stretch/aspect** = `"keep"`

There is currently no description for this property. Please help us by contributing one!

String **display/window/stretch/mode** = `"disabled"`

Defines how the base size is stretched to fit the resolution of the window or screen.

"disabled": No stretching happens. One unit in the scene corresponds to one pixel on the screen. In this mode, display/window/stretch/aspect has no effect. Recommended for non-game applications.

"canvas_items": The base size specified in width and height in the project settings is stretched to cover the whole screen (taking display/window/stretch/aspect into account). This means that everything is rendered directly at the target resolution. 3D is unaffected, while in 2D, there is no longer a 1:1 correspondence between sprite pixels and screen pixels, which may result in scaling artifacts. Recommended for most games that don't use a pixel art aesthetic, although it is possible to use this stretch mode for pixel art games too (especially in 3D).

"viewport": The size of the root Viewport is set precisely to the base size specified in the Project Settings' Display section. The scene is rendered to this viewport first. Finally, this viewport is scaled to fit the screen (taking display/window/stretch/aspect into account). Recommended for games that use a pixel art aesthetic.

---

float **display/window/stretch/scale** = `1.0`

The scale factor multiplier to use for 2D elements. This multiplies the final scale factor determined by display/window/stretch/mode. If using the **Disabled** stretch mode, this scale factor is applied as-is. This can be adjusted to make the UI easier to read on certain displays.

String **display/window/stretch/scale_mode** = `"fractional"` 🔗

The policy to use to determine the final scale factor for 2D elements. This affects how display/window/stretch/scale is applied, in addition to the automatic scale factor determined by display/window/stretch/mode.

"fractional": The scale factor will not be modified.

"integer": The scale factor will be floored to an integer value, which means that the screen size will always be an integer multiple of the base viewport size. This provides a crisp pixel art appearance.

Note: When using integer scaling with a stretch mode, resizing the window to be smaller than the base viewport size will clip the contents. Consider preventing that by setting Window.min_size to the same value as the base viewport size defined in display/window/size/viewport_width and display/window/size/viewport_height.

bool **display/window/subwindows/embed_subwindows** = `true` 🔗

If `true`, subwindows are embedded in the main window (this is also called single-window mode). Single-window mode can be faster as it does not need to create a separate window for every popup and tooltip, which can be a slow operation depending on the operating system and rendering method in use.

If `false`, subwindows are created as separate windows (this is also called multi-window mode). This allows them to be moved outside the main window and use native operating system window decorations.

This is equivalent to EditorSettings.interface/editor/single_window_mode in the editor, except the setting's value is inverted.

int **display/window/vsync/vsync_mode** = `1` 🔗

Sets the V-Sync mode for the main game window. The editor's own V-Sync mode can be set using EditorSettings.interface/editor/vsync_mode.

See VSyncMode for possible values and how they affect the behavior of your application.

Depending on the platform and rendering method, the engine will fall back to **Enabled** if the desired mode is not supported.

V-Sync can be disabled on the command line using the `--disable-vsync` command line argument.

Note: The **Adaptive** and **Mailbox** V-Sync modes are only supported in the Forward+ and Mobile rendering methods, not Compatibility.

Note: This property is only read when the project starts. To change the V-Sync mode at runtime, call DisplayServer.window_set_vsync_mode instead.

String **dotnet/project/assembly_name** = `""` 🔗

Name of the .NET assembly. This name is used as the name of the `.csproj` and `.sln` files. By default, it's set to the name of the project (application/config/name) allowing to change it in the future without affecting the .NET assembly.

---

int **dotnet/project/assembly_reload_attempts** = `3` 🔗

Number of times to attempt assembly reloading after rebuilding .NET assemblies. Effectively also the timeout in seconds to wait for unloading of script assemblies to finish.

String **dotnet/project/solution_directory** = `""` 🔗

Directory that contains the `.sln` file. By default, the `.sln` files is in the root of the project directory, next to the `project.godot` and `.csproj` files.

Changing this value allows setting up a multi-project scenario where there are multiple `.csproj`. Keep in mind that the Godot project is considered one of the C# projects in the workspace and it's root directory should contain the `project.godot` and `.csproj` next to each other.

---

bool **editor/export/convert_text_resources_to_binary** = `true` 🔗

If `true`, text resource (`tres`) and text scene (`tscn`) files are converted to their corresponding binary format on export. This decreases file sizes and speeds up loading slightly.

Note: Because a resource's file extension may change in an exported project, it is heavily recommended to use @GDScript.load or ResourceLoader instead of FileAccess to load resources dynamically.

Note: The project settings file (`project.godot`) will always be converted to binary on export, regardless of this setting.

---

int **editor/import/atlas_max_width** = `2048` 🔗

The maximum width to use when importing textures as an atlas. The value will be rounded to the nearest power of two when used. Use this to prevent imported textures from growing too large in the other direction.

---

bool **editor/import/reimport_missing_imported_files** = `true` 🔗

There is currently no description for this property. Please help us by contributing one!

---

bool **editor/import/use_multiple_threads** = `true` 🔗

If `true` importing of resources is run on multiple threads.

---

bool **editor/movie_writer/disable_vsync** = `false` 🔗

If `true`, requests V-Sync to be disabled when writing a movie (similar to setting display/window/vsync/vsync_mode to **Disabled**). This can speed up video writing if the hardware is fast enough to render, encode and save the video at a framerate higher than the monitor's refresh rate.

Note: editor/movie_writer/disable_vsync has no effect if the operating system or graphics driver forces V-Sync with no way for applications to disable it.

---

int **editor/movie_writer/fps** = `60` 🔗

The number of frames per second to record in the video when writing a movie. Simulation speed will adjust to always match the specified framerate, which means the engine will appear to run slower at higher editor/movie_writer/fps values. Certain FPS values will require you to adjust editor/movie_writer/mix_rate to prevent audio from desynchronizing over time.

This can be specified manually on the command line using the `--fixed-fps <fps>` command line argument.

---

int **editor/movie_writer/mix_rate** = `48000` 🔗

The audio mix rate to use in the recorded audio when writing a movie (in Hz). This can be different from audio/driver/mix_rate, but this value must be divisible by editor/movie_writer/fps to prevent audio from desynchronizing over time.

---

float **editor/movie_writer/mjpeg_quality** = `0.75` 🔗

The JPEG quality to use when writing a video to an AVI file, between `0.01` and `1.0` (inclusive). Higher `quality` values result in better-looking output at the cost of larger file sizes. Recommended `quality` values are between `0.75` and `0.9`. Even at quality `1.0`, JPEG compression remains lossy.

Note: This does not affect the audio quality or writing PNG image sequences.

String **editor/movie_writer/movie_file** = `""` 🔗

The output path for the movie. The file extension determines the MovieWriter that will be used.

Godot has 2 built-in MovieWriters:

- AVI container with MJPEG for video and uncompressed audio (`.avi` file extension). Lossy compression, medium file sizes, fast encoding. The lossy compression quality can be adjusted by changing editor/movie_writer/mjpeg_quality. The resulting file can be viewed in most video players, but it must be converted to another format for viewing on the web or by Godot with VideoStreamPlayer. MJPEG does not support transparency. AVI output is currently limited to a file of 4 GB in size at most.
- PNG image sequence for video and WAV for audio (`.png` file extension). Lossless compression, large file sizes, slow encoding. Designed to be encoded to a video file with another tool such as FFmpeg [https://ffmpeg.org/] after recording. Transparency is currently not supported, even if the root viewport is set to be transparent.

If you need to encode to a different format or pipe a stream through third-party software, you can extend this MovieWriter class to create your own movie writers.

When using PNG output, the frame number will be appended at the end of the file name. It starts from 0 and is padded with 8 digits to ensure correct sorting and easier processing. For example, if the output path is `/tmp/hello.png`, the first two frames will be `/tmp/hello00000000.png` and `/tmp/hello00000001.png`. The audio will be saved at `/tmp/hello.wav`.

---

int **editor/movie_writer/speaker_mode** = `0` 🔗

The speaker mode to use in the recorded audio when writing a movie. See SpeakerMode for possible values.

String **editor/naming/default_signal_callback_name** = `"_on_{node_name}_{signal_name}"` 🔗

The format of the default signal callback name (in the Signal Connection Dialog). The following substitutions are available: `{NodeName}`, `{nodeName}`, `{node_name}`, `{SignalName}`, `{signalName}`, and `{signal_name}`.

String **editor/naming/default_signal_callback_to_self_name** = `"_on_{signal_name}"` 🔗

The format of the default signal callback name when a signal connects to the same node that emits it (in the Signal Connection Dialog). The following substitutions are available: `{NodeName}`, `{nodeName}`, `{node_name}`, `{SignalName}`, `{signalName}`, and `{signal_name}`.

---

int **editor/naming/node_name_casing** = `0` 🔗

When creating node names automatically, set the type of casing to use in this project. This is mostly an editor setting.

---

int **editor/naming/node_name_num_separator** = `0` 🔗

What to use to separate node name from number. This is mostly an editor setting.

---

int **editor/naming/scene_name_casing** = `2` 🔗

When generating scene file names from scene root node, set the type of casing to use in this project. This is mostly an editor setting.

---

int **editor/naming/script_name_casing** = `0` 🔗

When generating script file names from the selected node, set the type of casing to use in this project. This is mostly an editor setting.

String **editor/run/main_run_args** = `""` 🔗

The command-line arguments to append to Godot's own command line when running the project. This doesn't affect the editor itself.

It is possible to make another executable run Godot by using the `%command%` placeholder. The placeholder will be replaced with Godot's own command line. Program-specific arguments should be placed *before* the placeholder, whereas Godot-specific arguments should be placed *after* the placeholder.

For example, this can be used to force the project to run on the dedicated GPU in an NVIDIA Optimus system on Linux:

```
prime-run %command%
```

PackedStringArray **editor/script/search_in_file_extensions** 🔗

Text-based file extensions to include in the script editor's "Find in Files" feature. You can add e.g. `tscn` if you wish to also parse your scene files, especially if you use built-in scripts which are serialized in the scene files.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedStringArray for more details.

String **editor/script/templates_search_path** = `"res://script_templates"` 🔗

Search path for project-specific script templates. Godot will search for script templates both in the editor-specific path and in this project-specific path.

---

bool **editor/version_control/autoload_on_startup** = `false` 🔗

There is currently no description for this property. Please help us by contributing one!

String **editor/version_control/plugin_name** = `""` 🔗

There is currently no description for this property. Please help us by contributing one!

---

bool **filesystem/import/blender/enabled** = `true` 🔗

If `true`, Blender 3D scene files with the `.blend` extension will be imported by converting them to glTF 2.0.

This requires configuring a path to a Blender executable in the editor settings at `filesystem/import/blender/blender_path`. Blender 3.0 or later is required.

---

bool **filesystem/import/blender/enabled.android** = `false` 🔗

Override for filesystem/import/blender/enabled on Android where Blender can't easily be accessed from Godot.

---

bool **filesystem/import/blender/enabled.web** = `false` 🔗

Override for filesystem/import/blender/enabled on the Web where Blender can't easily be accessed from Godot.

---

bool **filesystem/import/fbx2gltf/enabled** = `true`

If `true`, Autodesk FBX 3D scene files with the `.fbx` extension will be imported by converting them to glTF 2.0.

This requires configuring a path to an FBX2glTF executable in the editor settings at EditorSettings.filesystem/import/fbx/fbx2gltf_path.

---

bool **filesystem/import/fbx2gltf/enabled.android** = `false`

Override for filesystem/import/fbx2gltf/enabled on Android where FBX2glTF can't easily be accessed from Godot.

---

bool **filesystem/import/fbx2gltf/enabled.web** = `false`

Override for filesystem/import/fbx2gltf/enabled on the Web where FBX2glTF can't easily be accessed from Godot.

---

int **gui/common/default_scroll_deadzone** = `0`

Default value for ScrollContainer.scroll_deadzone, which will be used for all ScrollContainers unless overridden.

---

bool **gui/common/snap_controls_to_pixels** = `true`

If `true`, snaps Control node vertices to the nearest pixel to ensure they remain crisp even when the camera moves or zooms.

---

bool **gui/common/swap_cancel_ok**

If `true`, swaps **Cancel** and **OK** buttons in dialogs on Windows to follow interface conventions. DisplayServer.get_swap_cancel_ok can be used to query whether buttons are swapped at run-time.

Note: This doesn't affect native dialogs such as the ones spawned by DisplayServer.dialog_show.

---

int **gui/common/text_edit_undo_stack_max_size** = `1024`

Maximum undo/redo history size for TextEdit fields.

---

bool **gui/fonts/dynamic_fonts/use_oversampling** = `true`

There is currently no description for this property. Please help us by contributing one!

String **gui/theme/custom** = `""`

Path to a custom Theme resource file to use for the project (`.theme` or generic `.tres`/`.res` extension).

String **gui/theme/custom_font** = `""`

Path to a custom Font resource to use as default for all GUI elements of the project.

int **gui/theme/default_font_antialiasing** = `1`

Font anti-aliasing mode for the default project font. See FontFile.antialiasing.

Note: This setting does not affect custom Fonts used within the project. Use the **Import** dock for that instead (see ResourceImporterDynamicFont.antialiasing).

bool **gui/theme/default_font_generate_mipmaps** = `false`

If set to `true`, the default font will have mipmaps generated. This prevents text from looking grainy when a Control is scaled down, or when a Label3D is viewed from a long distance (if Label3D.texture_filter is set to a mode that displays mipmaps).

Enabling gui/theme/default_font_generate_mipmaps increases font generation time and memory usage. Only enable this setting if you actually need it.

Note: This setting does not affect custom Fonts used within the project. Use the **Import** dock for that instead (see ResourceImporterDynamicFont.generate_mipmaps).

int **gui/theme/default_font_hinting** = `1`

Font hinting mode for the default project font. See FontFile.hinting.

Note: This setting does not affect custom Fonts used within the project. Use the **Import** dock for that instead (see ResourceImporterDynamicFont.hinting).

bool **gui/theme/default_font_multichannel_signed_distance_field** = `false`

If set to `true`, the default font will use multichannel signed distance field (MSDF) for crisp rendering at any size. Since this approach does not rely on rasterizing the font every time its size changes, this allows for resizing the font in real-time without any performance penalty. Text will also not look grainy for Controls that are scaled down (or for Label3Ds viewed from a long distance).

MSDF font rendering can be combined with gui/theme/default_font_generate_mipmaps to further improve font rendering quality when scaled down.

Note: This setting does not affect custom Fonts used within the project. Use the **Import** dock for that instead (see ResourceImporterDynamicFont.multichannel_signed_distance_field).

int **gui/theme/default_font_subpixel_positioning** = `1`

Font glyph subpixel positioning mode for the default project font. See FontFile.subpixel_positioning.

Note: This setting does not affect custom Fonts used within the project. Use the **Import** dock for that instead (see ResourceImporterDynamicFont.subpixel_positioning).

float **gui/theme/default_theme_scale** = `1.0` 🔗

The default scale factor for Controls, when not overridden by a Theme.

Note: This property is only read when the project starts. To change the default scale at runtime, set ThemeDB.fallback_base_scale instead.

---

int **gui/theme/lcd_subpixel_layout** = `1` 🔗

LCD subpixel layout used for font anti-aliasing. See FontLCDSubpixelLayout.

---

float **gui/timers/button_shortcut_feedback_highlight_time** = `0.2` 🔗

When BaseButton.shortcut_feedback is enabled, this is the time the BaseButton will remain highlighted after a shortcut.

---

int **gui/timers/incremental_search_max_interval_msec** = `2000` 🔗

Timer setting for incremental search in Tree, ItemList, etc. controls (in milliseconds).

---

float **gui/timers/text_edit_idle_detect_sec** = `3` 🔗

Timer for detecting idle in TextEdit (in seconds).

---

float **gui/timers/tooltip_delay_sec** = `0.5` 🔗

Default delay for tooltips (in seconds).

---

float **gui/timers/tooltip_delay_sec.editor_hint** = `0.5` 🔗

Delay for tooltips in the editor.

Dictionary **input/ui_accept** 🔗

Default InputEventAction to confirm a focused button, menu or list item, or validate input.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_cancel** 🔗

Default InputEventAction to discard a modal or pending input.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_copy** 🔗

Default InputEventAction to copy a selection to the clipboard.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_cut**

Default InputEventAction to cut a selection to the clipboard.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_down**

Default InputEventAction to move down in the UI.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_end**

Default InputEventAction to go to the end position of a Control (e.g. last item in an ItemList or a Tree), matching the behavior of @GlobalScope.KEY_END on typical desktop UI systems.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_filedialog_refresh**

Default InputEventAction to refresh the contents of the current directory of a FileDialog.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_filedialog_show_hidden**

Default InputEventAction to toggle showing hidden files and directories in a FileDialog.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_filedialog_up_one_level**

Default InputEventAction to go up one directory in a FileDialog.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_focus_next**

Default InputEventAction to focus the next Control in the scene. The focus behavior can be configured via Control.focus_next.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_focus_prev** 🔗

Default InputEventAction to focus the previous Control in the scene. The focus behavior can be configured via Control.focus_previous.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_graph_delete** 🔗

Default InputEventAction to delete a GraphNode in a GraphEdit.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_graph_duplicate** 🔗

Default InputEventAction to duplicate a GraphNode in a GraphEdit.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_home** 🔗

Default InputEventAction to go to the start position of a Control (e.g. first item in an ItemList or a Tree), matching the behavior of @GlobalScope.KEY_HOME on typical desktop UI systems.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_left** 🔗

Default InputEventAction to move left in the UI.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_menu** 🔗

Default InputEventAction to open a context menu in a text field.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_page_down** 🔗

Default InputEventAction to go down a page in a Control (e.g. in an ItemList or a Tree), matching the behavior of @GlobalScope.KEY_PAGEDOWN on typical desktop UI systems.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_page_up** 🔗

Default InputEventAction to go up a page in a Control (e.g. in an ItemList or a Tree), matching the behavior of @GlobalScope.KEY_PAGEUP on typical desktop UI systems.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_paste** 🔗

Default InputEventAction to paste from the clipboard.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_redo** 🔗

Default InputEventAction to redo an undone action.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_right** 🔗

Default InputEventAction to move right in the UI.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_select** 🔗

Default InputEventAction to select an item in a Control (e.g. in an ItemList or a Tree).

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_swap_input_direction** 🔗

Default InputEventAction to swap input direction, i.e. change between left-to-right to right-to-left modes. Affects text-editing controls (LineEdit, TextEdit).

Dictionary **input/ui_text_add_selection_for_next_occurrence** 🔗

If a selection is currently active with the last caret in text fields, searches for the next occurrence of the selection, adds a caret and selects the next occurrence.

If no selection is currently active with the last caret in text fields, selects the word currently under the caret.

The action can be performed sequentially for all occurrences of the selection of the last caret and for all existing carets.

The viewport is adjusted to the latest newly added caret.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_backspace** 🔗

Default InputEventAction to delete the character before the text cursor.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_backspace_all_to_left** 🔗

Default InputEventAction to delete **all** text before the text cursor.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_backspace_all_to_left.macos** 🔗

macOS specific override for the shortcut to delete all text before the text cursor.

Dictionary **input/ui_text_backspace_word** 🔗

Default InputEventAction to delete all characters before the cursor up until a whitespace or punctuation character.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_backspace_word.macos** 🔗

macOS specific override for the shortcut to delete a word.

Dictionary **input/ui_text_caret_add_above** 🔗

Default InputEventAction to add an additional caret above every caret of a text.

Dictionary **input/ui_text_caret_add_above.macos**

macOS specific override for the shortcut to add a caret above every caret.

Dictionary **input/ui_text_caret_add_below**

Default InputEventAction to add an additional caret below every caret of a text.

Dictionary **input/ui_text_caret_add_below.macos**

macOS specific override for the shortcut to add a caret below every caret.

Dictionary **input/ui_text_caret_document_end**

Default InputEventAction to move the text cursor to the end of the text.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_document_end.macos**

macOS specific override for the shortcut to move the text cursor to the end of the text.

Dictionary **input/ui_text_caret_document_start**

Default InputEventAction to move the text cursor to the start of the text.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_document_start.macos**

macOS specific override for the shortcut to move the text cursor to the start of the text.

Dictionary **input/ui_text_caret_down**

Default InputEventAction to move the text cursor down.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_left**

Default InputEventAction to move the text cursor left.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_line_end** 🔗

Default InputEventAction to move the text cursor to the end of the line.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_line_end.macos** 🔗

macOS specific override for the shortcut to move the text cursor to the end of the line.

Dictionary **input/ui_text_caret_line_start** 🔗

Default InputEventAction to move the text cursor to the start of the line.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_line_start.macos** 🔗

macOS specific override for the shortcut to move the text cursor to the start of the line.

Dictionary **input/ui_text_caret_page_down** 🔗

Default InputEventAction to move the text cursor down one page.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_page_up** 🔗

Default InputEventAction to move the text cursor up one page.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_right** 🔗

Default InputEventAction to move the text cursor right.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_up** 🔗

Default InputEventAction to move the text cursor up.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_word_left** 🔗

Default InputEventAction to move the text cursor left to the next whitespace or punctuation.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_word_left.macos** 🔗

macOS specific override for the shortcut to move the text cursor back one word.

Dictionary **input/ui_text_caret_word_right** 🔗

Default InputEventAction to move the text cursor right to the next whitespace or punctuation.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_caret_word_right.macos** 🔗

macOS specific override for the shortcut to move the text cursor forward one word.

Dictionary **input/ui_text_clear_carets_and_selection** 🔗

If there's only one caret active and with a selection, clears the selection.

In case there's more than one caret active, removes the secondary carets and clears their selections.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_completion_accept** 🔗

Default InputEventAction to accept an autocompletion hint.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_completion_query** 🔗

Default InputEventAction to request autocompletion.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_completion_replace** 🔗

Default InputEventAction to accept an autocompletion hint, replacing existing text.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_dedent** 🔗

Default InputEventAction to unindent text.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_delete** 🔗

Default InputEventAction to delete the character after the text cursor.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_delete_all_to_right** 🔗

Default InputEventAction to delete **all** text after the text cursor.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_delete_all_to_right.macos** 🔗

macOS specific override for the shortcut to delete all text after the text cursor.

Dictionary **input/ui_text_delete_word** 🔗

Default InputEventAction to delete all characters after the cursor up until a whitespace or punctuation character.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_delete_word.macos** 🔗

macOS specific override for the shortcut to delete a word after the text cursor.

Dictionary **input/ui_text_indent** 🔗

Default InputEventAction to indent the current line.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_newline** 🔗

Default InputEventAction to insert a new line at the position of the text cursor.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_newline_above** 🔗

Default InputEventAction to insert a new line before the current one.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_newline_blank** 🔗

Default InputEventAction to insert a new line after the current one.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_scroll_down** 🔗

Default InputEventAction to scroll down one line of text.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_scroll_down.macos** 🔗

macOS specific override for the shortcut to scroll down one line.

Dictionary **input/ui_text_scroll_up** 🔗

Default InputEventAction to scroll up one line of text.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_scroll_up.macos** 🔗

macOS specific override for the shortcut to scroll up one line.

Dictionary **input/ui_text_select_all** 🔗

Default InputEventAction to select all text.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_select_word_under_caret** 🔗

If no selection is currently active, selects the word currently under the caret in text fields. If a selection is currently active, deselects the current selection.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_select_word_under_caret.macos** 🔗

macOS specific override for the shortcut to select the word currently under the caret.

Dictionary **input/ui_text_skip_selection_for_next_occurrence** 🔗

If no selection is currently active with the last caret in text fields, searches for the next occurrence of the the word currently under the caret and moves the caret to the next occurrence. The action can be performed sequentially for other occurrences of the word under the last caret.

If a selection is currently active with the last caret in text fields, searches for the next occurrence of the selection, adds a caret, selects the next occurrence then deselects the previous selection and its associated caret. The action can be performed sequentially for other occurrences of the selection of the last caret.

The viewport is adjusted to the latest newly added caret.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_submit** 🔗

Default InputEventAction to submit a text field.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_text_toggle_insert_mode** 🔗

Default InputEventAction to toggle *insert mode* in a text field. While in insert mode, inserting new text overrides the character after the cursor, unless the next character is a new line.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_undo** 🔗

Default InputEventAction to undo the most recent action.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

Dictionary **input/ui_up**

Default InputEventAction to move up in the UI.

Note: Default `ui_*` actions cannot be removed as they are necessary for the internal logic of several Controls. The events assigned to the action can however be modified.

---

bool **input_devices/buffering/agile_event_flushing** = `false`

If `true`, key/touch/joystick events will be flushed just before every idle and physics frame.

If `false`, such events will be flushed only once per process frame, between iterations of the engine.

Enabling this can greatly improve the responsiveness to input, specially in devices that need to run multiple physics frames per visible (process) frame, because they can't run at the target frame rate.

Note: Currently implemented only on Android.

---

bool **input_devices/compatibility/legacy_just_pressed_behavior** = `false`

If `true`, Input.is_action_just_pressed and Input.is_action_just_released will only return `true` if the action is still in the respective state, i.e. an action that is pressed *and* released on the same frame will be missed.

If `false`, no input will be lost.

Note: You should in nearly all cases prefer the `false` setting. The legacy behavior is to enable supporting old projects that rely on the old logic, without changes to script.

String **input_devices/pen_tablet/driver**

Specifies the tablet driver to use. If left empty, the default driver will be used.

Note: The driver in use can be overridden at runtime via the `--tablet-driver` command line argument.

String **input_devices/pen_tablet/driver.windows**

Override for input_devices/pen_tablet/driver on Windows.

---

bool **input_devices/pointing/android/enable_long_press_as_right_click** = `false`

If `true`, long press events on an Android touchscreen are transformed into right click events.

---

bool **input_devices/pointing/android/enable_pan_and_scale_gestures** = `false`

If `true`, multi-touch pan and scale gestures are enabled on Android devices.

---

int **input_devices/pointing/android/rotary_input_scroll_axis** = `1`

On Wear OS devices, defines which axis of the mouse wheel rotary input is mapped to. This rotary input is usually performed by rotating the physical or virtual (touch-based) bezel on a smartwatch.

---

bool **input_devices/pointing/emulate_mouse_from_touch** = `true`

If `true`, sends mouse input events when tapping or swiping on the touchscreen.

---

bool **input_devices/pointing/emulate_touch_from_mouse** = `false`

If `true`, sends touch input events when clicking or dragging the mouse.

String **internationalization/locale/fallback** = `"en"`

The locale to fall back to if a translation isn't available in a given language. If left empty, `en` (English) will be used.

---

bool **internationalization/locale/include_text_server_data** = `false`

If `true`, text server break iteration rule sets, dictionaries and other optional data are included in the exported project.

Note: "ICU / HarfBuzz / Graphite" text server data includes dictionaries for Burmese, Chinese, Japanese, Khmer, Lao and Thai as well as Unicode Standard Annex #29 and Unicode Standard Annex #14 word and line breaking rules. Data is about 4 MB large.

Note: "Fallback" text server does not use additional data.

String **internationalization/locale/test** = `""`

If non-empty, this locale will be used when running the project from the editor.

---

bool **internationalization/pseudolocalization/double_vowels** = `false`

Double vowels in strings during pseudolocalization to simulate the lengthening of text due to localization.

---

float **internationalization/pseudolocalization/expansion_ratio** = `0.0`

The expansion ratio to use during pseudolocalization. A value of `0.3` is sufficient for most practical purposes, and will increase the length of each string by 30%.

---

bool **internationalization/pseudolocalization/fake_bidi** = `false`

If `true`, emulate bidirectional (right-to-left) text when pseudolocalization is enabled. This can be used to spot issues with RTL layout and UI mirroring that will crop up if the project is localized to RTL languages such as Arabic or Hebrew.

---

bool **internationalization/pseudolocalization/override** = `false` 🔗

Replace all characters in the string with `*`. Useful for finding non-localizable strings.

String **internationalization/pseudolocalization/prefix** = `"["` 🔗

Prefix that will be prepended to the pseudolocalized string.

---

bool **internationalization/pseudolocalization/replace_with_accents** = `true` 🔗

Replace all characters with their accented variants during pseudolocalization.

---

bool **internationalization/pseudolocalization/skip_placeholders** = `true` 🔗

Skip placeholders for string formatting like `%s` or `%f` during pseudolocalization. Useful to identify strings which need additional control characters to display correctly.

String **internationalization/pseudolocalization/suffix** = `"]"` 🔗

Suffix that will be appended to the pseudolocalized string.

---

bool **internationalization/pseudolocalization/use_pseudolocalization** = `false` 🔗

If `true`, enables pseudolocalization for the project. This can be used to spot untranslatable strings or layout issues that may occur once the project is localized to languages that have longer strings than the source language.

Note: This property is only read when the project starts. To toggle pseudolocalization at run-time, use TranslationServer.pseudolocalization_enabled instead.

---

bool **internationalization/rendering/force_right_to_left_layout_direction** = `false` 🔗

Force layout direction and text writing direction to RTL for all controls.

---

bool **internationalization/rendering/root_node_auto_translate** = `true` 🔗

If `true`, root node will use Node.AUTO_TRANSLATE_MODE_ALWAYS, otherwise Node.AUTO_TRANSLATE_MODE_DISABLED will be used.

Note: This property is only read when the project starts. To change the auto translate mode at runtime, set Node.auto_translate_mode of SceneTree.root instead.

---

int **internationalization/rendering/root_node_layout_direction** = `0` 🔗

Root node default layout direction.

String **internationalization/rendering/text_driver** = `""`

Specifies the TextServer to use. If left empty, the default will be used.

"ICU / HarfBuzz / Graphite" is the most advanced text driver, supporting right-to-left typesetting and complex scripts (for languages like Arabic, Hebrew, etc.). The "Fallback" text driver does not support right-to-left typesetting and complex scripts.

Note: The driver in use can be overridden at runtime via the `--text-driver` command line argument.

Note: There is an additional `Dummy` text driver available, which disables all text rendering and font-related functionality. This driver is not listed in the project settings, but it can be enabled when running the editor or project using the `--text-driver Dummy` command line argument.

String **layer_names/2d_navigation/layer_1** = `""`

Optional name for the 2D navigation layer 1. If left empty, the layer will display as "Layer 1".

String **layer_names/2d_navigation/layer_2** = `""`

Optional name for the 2D navigation layer 2. If left empty, the layer will display as "Layer 2".

String **layer_names/2d_navigation/layer_3** = `""`

Optional name for the 2D navigation layer 3. If left empty, the layer will display as "Layer 3".

String **layer_names/2d_navigation/layer_4** = `""`

Optional name for the 2D navigation layer 4. If left empty, the layer will display as "Layer 4".

String **layer_names/2d_navigation/layer_5** = `""`

Optional name for the 2D navigation layer 5. If left empty, the layer will display as "Layer 5".

String **layer_names/2d_navigation/layer_6** = `""`

Optional name for the 2D navigation layer 6. If left empty, the layer will display as "Layer 6".

String **layer_names/2d_navigation/layer_7** = `""`

Optional name for the 2D navigation layer 7. If left empty, the layer will display as "Layer 7".

String **layer_names/2d_navigation/layer_8** = `""`

Optional name for the 2D navigation layer 8. If left empty, the layer will display as "Layer 8".

String **layer_names/2d_navigation/layer_9** = `""`

Optional name for the 2D navigation layer 9. If left empty, the layer will display as "Layer 9".

String **layer_names/2d_navigation/layer_10** = `""`

Optional name for the 2D navigation layer 10. If left empty, the layer will display as "Layer 10".

String **layer_names/2d_navigation/layer_11** = `""`

Optional name for the 2D navigation layer 11. If left empty, the layer will display as "Layer 11".

String **layer_names/2d_navigation/layer_12** = `""`

Optional name for the 2D navigation layer 12. If left empty, the layer will display as "Layer 12".

String **layer_names/2d_navigation/layer_13** = `""`

Optional name for the 2D navigation layer 13. If left empty, the layer will display as "Layer 13".

String **layer_names/2d_navigation/layer_14** = `""`

Optional name for the 2D navigation layer 14. If left empty, the layer will display as "Layer 14".

String **layer_names/2d_navigation/layer_15** = `""`

Optional name for the 2D navigation layer 15. If left empty, the layer will display as "Layer 15".

String **layer_names/2d_navigation/layer_16** = `""`

Optional name for the 2D navigation layer 16. If left empty, the layer will display as "Layer 16".

String **layer_names/2d_navigation/layer_17** = `""`

Optional name for the 2D navigation layer 17. If left empty, the layer will display as "Layer 17".

String **layer_names/2d_navigation/layer_18** = `""`

Optional name for the 2D navigation layer 18. If left empty, the layer will display as "Layer 18".

String **layer_names/2d_navigation/layer_19** = `""`

Optional name for the 2D navigation layer 19. If left empty, the layer will display as "Layer 19".

String **layer_names/2d_navigation/layer_20** = `""`

Optional name for the 2D navigation layer 20. If left empty, the layer will display as "Layer 20".

String **layer_names/2d_navigation/layer_21** = ""

Optional name for the 2D navigation layer 21. If left empty, the layer will display as "Layer 21".

String **layer_names/2d_navigation/layer_22** = ""

Optional name for the 2D navigation layer 22. If left empty, the layer will display as "Layer 22".

String **layer_names/2d_navigation/layer_23** = ""

Optional name for the 2D navigation layer 23. If left empty, the layer will display as "Layer 23".

String **layer_names/2d_navigation/layer_24** = ""

Optional name for the 2D navigation layer 24. If left empty, the layer will display as "Layer 24".

String **layer_names/2d_navigation/layer_25** = ""

Optional name for the 2D navigation layer 25. If left empty, the layer will display as "Layer 25".

String **layer_names/2d_navigation/layer_26** = ""

Optional name for the 2D navigation layer 26. If left empty, the layer will display as "Layer 26".

String **layer_names/2d_navigation/layer_27** = ""

Optional name for the 2D navigation layer 27. If left empty, the layer will display as "Layer 27".

String **layer_names/2d_navigation/layer_28** = ""

Optional name for the 2D navigation layer 28. If left empty, the layer will display as "Layer 28".

String **layer_names/2d_navigation/layer_29** = ""

Optional name for the 2D navigation layer 29. If left empty, the layer will display as "Layer 29".

String **layer_names/2d_navigation/layer_30** = ""

Optional name for the 2D navigation layer 30. If left empty, the layer will display as "Layer 30".

String **layer_names/2d_navigation/layer_31** = ""

Optional name for the 2D navigation layer 31. If left empty, the layer will display as "Layer 31".

String **layer_names/2d_navigation/layer_32** = `""`

Optional name for the 2D navigation layer 32. If left empty, the layer will display as "Layer 32".

String **layer_names/2d_physics/layer_1** = `""`

Optional name for the 2D physics layer 1. If left empty, the layer will display as "Layer 1".

String **layer_names/2d_physics/layer_2** = `""`

Optional name for the 2D physics layer 2. If left empty, the layer will display as "Layer 2".

String **layer_names/2d_physics/layer_3** = `""`

Optional name for the 2D physics layer 3. If left empty, the layer will display as "Layer 3".

String **layer_names/2d_physics/layer_4** = `""`

Optional name for the 2D physics layer 4. If left empty, the layer will display as "Layer 4".

String **layer_names/2d_physics/layer_5** = `""`

Optional name for the 2D physics layer 5. If left empty, the layer will display as "Layer 5".

String **layer_names/2d_physics/layer_6** = `""`

Optional name for the 2D physics layer 6. If left empty, the layer will display as "Layer 6".

String **layer_names/2d_physics/layer_7** = `""`

Optional name for the 2D physics layer 7. If left empty, the layer will display as "Layer 7".

String **layer_names/2d_physics/layer_8** = `""`

Optional name for the 2D physics layer 8. If left empty, the layer will display as "Layer 8".

String **layer_names/2d_physics/layer_9** = `""`

Optional name for the 2D physics layer 9. If left empty, the layer will display as "Layer 9".

String **layer_names/2d_physics/layer_10** = `""`

Optional name for the 2D physics layer 10. If left empty, the layer will display as "Layer 10".

String **layer_names/2d_physics/layer_11** = `""`

Optional name for the 2D physics layer 11. If left empty, the layer will display as "Layer 11".

String **layer_names/2d_physics/layer_12** = `""`

Optional name for the 2D physics layer 12. If left empty, the layer will display as "Layer 12".

String **layer_names/2d_physics/layer_13** = `""`

Optional name for the 2D physics layer 13. If left empty, the layer will display as "Layer 13".

String **layer_names/2d_physics/layer_14** = `""`

Optional name for the 2D physics layer 14. If left empty, the layer will display as "Layer 14".

String **layer_names/2d_physics/layer_15** = `""`

Optional name for the 2D physics layer 15. If left empty, the layer will display as "Layer 15".

String **layer_names/2d_physics/layer_16** = `""`

Optional name for the 2D physics layer 16. If left empty, the layer will display as "Layer 16".

String **layer_names/2d_physics/layer_17** = `""`

Optional name for the 2D physics layer 17. If left empty, the layer will display as "Layer 17".

String **layer_names/2d_physics/layer_18** = `""`

Optional name for the 2D physics layer 18. If left empty, the layer will display as "Layer 18".

String **layer_names/2d_physics/layer_19** = `""`

Optional name for the 2D physics layer 19. If left empty, the layer will display as "Layer 19".

String **layer_names/2d_physics/layer_20** = `""`

Optional name for the 2D physics layer 20. If left empty, the layer will display as "Layer 20".

String **layer_names/2d_physics/layer_21** = `""`

Optional name for the 2D physics layer 21. If left empty, the layer will display as "Layer 21".

String **layer_names/2d_physics/layer_22** = `""`

Optional name for the 2D physics layer 22. If left empty, the layer will display as "Layer 22".

String **layer_names/2d_physics/layer_23** = ""

Optional name for the 2D physics layer 23. If left empty, the layer will display as "Layer 23".

String **layer_names/2d_physics/layer_24** = ""

Optional name for the 2D physics layer 24. If left empty, the layer will display as "Layer 24".

String **layer_names/2d_physics/layer_25** = ""

Optional name for the 2D physics layer 25. If left empty, the layer will display as "Layer 25".

String **layer_names/2d_physics/layer_26** = ""

Optional name for the 2D physics layer 26. If left empty, the layer will display as "Layer 26".

String **layer_names/2d_physics/layer_27** = ""

Optional name for the 2D physics layer 27. If left empty, the layer will display as "Layer 27".

String **layer_names/2d_physics/layer_28** = ""

Optional name for the 2D physics layer 28. If left empty, the layer will display as "Layer 28".

String **layer_names/2d_physics/layer_29** = ""

Optional name for the 2D physics layer 29. If left empty, the layer will display as "Layer 29".

String **layer_names/2d_physics/layer_30** = ""

Optional name for the 2D physics layer 30. If left empty, the layer will display as "Layer 30".

String **layer_names/2d_physics/layer_31** = ""

Optional name for the 2D physics layer 31. If left empty, the layer will display as "Layer 31".

String **layer_names/2d_physics/layer_32** = ""

Optional name for the 2D physics layer 32. If left empty, the layer will display as "Layer 32".

String **layer_names/2d_render/layer_1** = ""

Optional name for the 2D render layer 1. If left empty, the layer will display as "Layer 1".

String **layer_names/2d_render/layer_2** = `""`

Optional name for the 2D render layer 2. If left empty, the layer will display as "Layer 2".

String **layer_names/2d_render/layer_3** = `""`

Optional name for the 2D render layer 3. If left empty, the layer will display as "Layer 3".

String **layer_names/2d_render/layer_4** = `""`

Optional name for the 2D render layer 4. If left empty, the layer will display as "Layer 4".

String **layer_names/2d_render/layer_5** = `""`

Optional name for the 2D render layer 5. If left empty, the layer will display as "Layer 5".

String **layer_names/2d_render/layer_6** = `""`

Optional name for the 2D render layer 6. If left empty, the layer will display as "Layer 6".

String **layer_names/2d_render/layer_7** = `""`

Optional name for the 2D render layer 7. If left empty, the layer will display as "Layer 7".

String **layer_names/2d_render/layer_8** = `""`

Optional name for the 2D render layer 8. If left empty, the layer will display as "Layer 8".

String **layer_names/2d_render/layer_9** = `""`

Optional name for the 2D render layer 9. If left empty, the layer will display as "Layer 9".

String **layer_names/2d_render/layer_10** = `""`

Optional name for the 2D render layer 10. If left empty, the layer will display as "Layer 10".

String **layer_names/2d_render/layer_11** = `""`

Optional name for the 2D render layer 11. If left empty, the layer will display as "Layer 11".

String **layer_names/2d_render/layer_12** = `""`

Optional name for the 2D render layer 12. If left empty, the layer will display as "Layer 12".

String **layer_names/2d_render/layer_13** = `""`

Optional name for the 2D render layer 13. If left empty, the layer will display as "Layer 13".

String **layer_names/2d_render/layer_14** = ""

Optional name for the 2D render layer 14. If left empty, the layer will display as "Layer 14".

String **layer_names/2d_render/layer_15** = ""

Optional name for the 2D render layer 15. If left empty, the layer will display as "Layer 15".

String **layer_names/2d_render/layer_16** = ""

Optional name for the 2D render layer 16. If left empty, the layer will display as "Layer 16".

String **layer_names/2d_render/layer_17** = ""

Optional name for the 2D render layer 17. If left empty, the layer will display as "Layer 17".

String **layer_names/2d_render/layer_18** = ""

Optional name for the 2D render layer 18. If left empty, the layer will display as "Layer 18".

String **layer_names/2d_render/layer_19** = ""

Optional name for the 2D render layer 19. If left empty, the layer will display as "Layer 19".

String **layer_names/2d_render/layer_20** = ""

Optional name for the 2D render layer 20. If left empty, the layer will display as "Layer 20".

String **layer_names/3d_navigation/layer_1** = ""

Optional name for the 3D navigation layer 1. If left empty, the layer will display as "Layer 1".

String **layer_names/3d_navigation/layer_2** = ""

Optional name for the 3D navigation layer 2. If left empty, the layer will display as "Layer 2".

String **layer_names/3d_navigation/layer_3** = ""

Optional name for the 3D navigation layer 3. If left empty, the layer will display as "Layer 3".

String **layer_names/3d_navigation/layer_4** = ""

Optional name for the 3D navigation layer 4. If left empty, the layer will display as "Layer 4".

String **layer_names/3d_navigation/layer_5** = `""` 🔗

Optional name for the 3D navigation layer 5. If left empty, the layer will display as "Layer 5".

String **layer_names/3d_navigation/layer_6** = `""` 🔗

Optional name for the 3D navigation layer 6. If left empty, the layer will display as "Layer 6".

String **layer_names/3d_navigation/layer_7** = `""` 🔗

Optional name for the 3D navigation layer 7. If left empty, the layer will display as "Layer 7".

String **layer_names/3d_navigation/layer_8** = `""` 🔗

Optional name for the 3D navigation layer 8. If left empty, the layer will display as "Layer 8".

String **layer_names/3d_navigation/layer_9** = `""` 🔗

Optional name for the 3D navigation layer 9. If left empty, the layer will display as "Layer 9".

String **layer_names/3d_navigation/layer_10** = `""` 🔗

Optional name for the 3D navigation layer 10. If left empty, the layer will display as "Layer 10".

String **layer_names/3d_navigation/layer_11** = `""` 🔗

Optional name for the 3D navigation layer 11. If left empty, the layer will display as "Layer 11".

String **layer_names/3d_navigation/layer_12** = `""` 🔗

Optional name for the 3D navigation layer 12. If left empty, the layer will display as "Layer 12".

String **layer_names/3d_navigation/layer_13** = `""` 🔗

Optional name for the 3D navigation layer 13. If left empty, the layer will display as "Layer 13".

String **layer_names/3d_navigation/layer_14** = `""` 🔗

Optional name for the 3D navigation layer 14. If left empty, the layer will display as "Layer 14".

String **layer_names/3d_navigation/layer_15** = `""` 🔗

Optional name for the 3D navigation layer 15. If left empty, the layer will display as "Layer 15".

String **layer_names/3d_navigation/layer_16** = ""

Optional name for the 3D navigation layer 16. If left empty, the layer will display as "Layer 16".

String **layer_names/3d_navigation/layer_17** = ""

Optional name for the 3D navigation layer 17. If left empty, the layer will display as "Layer 17".

String **layer_names/3d_navigation/layer_18** = ""

Optional name for the 3D navigation layer 18. If left empty, the layer will display as "Layer 18".

String **layer_names/3d_navigation/layer_19** = ""

Optional name for the 3D navigation layer 19. If left empty, the layer will display as "Layer 19".

String **layer_names/3d_navigation/layer_20** = ""

Optional name for the 3D navigation layer 20. If left empty, the layer will display as "Layer 20".

String **layer_names/3d_navigation/layer_21** = ""

Optional name for the 3D navigation layer 21. If left empty, the layer will display as "Layer 21".

String **layer_names/3d_navigation/layer_22** = ""

Optional name for the 3D navigation layer 22. If left empty, the layer will display as "Layer 22".

String **layer_names/3d_navigation/layer_23** = ""

Optional name for the 3D navigation layer 23. If left empty, the layer will display as "Layer 23".

String **layer_names/3d_navigation/layer_24** = ""

Optional name for the 3D navigation layer 24. If left empty, the layer will display as "Layer 24".

String **layer_names/3d_navigation/layer_25** = ""

Optional name for the 3D navigation layer 25. If left empty, the layer will display as "Layer 25".

String **layer_names/3d_navigation/layer_26** = ""

Optional name for the 3D navigation layer 26. If left empty, the layer will display as "Layer 26".

String **layer_names/3d_navigation/layer_27** = ""

Optional name for the 3D navigation layer 27. If left empty, the layer will display as "Layer 27".

String **layer_names/3d_navigation/layer_28** = `""`

Optional name for the 3D navigation layer 28. If left empty, the layer will display as "Layer 28".

String **layer_names/3d_navigation/layer_29** = `""`

Optional name for the 3D navigation layer 29. If left empty, the layer will display as "Layer 29".

String **layer_names/3d_navigation/layer_30** = `""`

Optional name for the 3D navigation layer 30. If left empty, the layer will display as "Layer 30".

String **layer_names/3d_navigation/layer_31** = `""`

Optional name for the 3D navigation layer 31. If left empty, the layer will display as "Layer 31".

String **layer_names/3d_navigation/layer_32** = `""`

Optional name for the 3D navigation layer 32. If left empty, the layer will display as "Layer 32".

String **layer_names/3d_physics/layer_1** = `""`

Optional name for the 3D physics layer 1. If left empty, the layer will display as "Layer 1".

String **layer_names/3d_physics/layer_2** = `""`

Optional name for the 3D physics layer 2. If left empty, the layer will display as "Layer 2".

String **layer_names/3d_physics/layer_3** = `""`

Optional name for the 3D physics layer 3. If left empty, the layer will display as "Layer 3".

String **layer_names/3d_physics/layer_4** = `""`

Optional name for the 3D physics layer 4. If left empty, the layer will display as "Layer 4".

String **layer_names/3d_physics/layer_5** = `""`

Optional name for the 3D physics layer 5. If left empty, the layer will display as "Layer 5".

String **layer_names/3d_physics/layer_6** = `""`

Optional name for the 3D physics layer 6. If left empty, the layer will display as "Layer 6".

String **layer_names/3d_physics/layer_7** = `""` 🔗

Optional name for the 3D physics layer 7. If left empty, the layer will display as "Layer 7".

String **layer_names/3d_physics/layer_8** = `""` 🔗

Optional name for the 3D physics layer 8. If left empty, the layer will display as "Layer 8".

String **layer_names/3d_physics/layer_9** = `""` 🔗

Optional name for the 3D physics layer 9. If left empty, the layer will display as "Layer 9".

String **layer_names/3d_physics/layer_10** = `""` 🔗

Optional name for the 3D physics layer 10. If left empty, the layer will display as "Layer 10".

String **layer_names/3d_physics/layer_11** = `""` 🔗

Optional name for the 3D physics layer 11. If left empty, the layer will display as "Layer 11".

String **layer_names/3d_physics/layer_12** = `""` 🔗

Optional name for the 3D physics layer 12. If left empty, the layer will display as "Layer 12".

String **layer_names/3d_physics/layer_13** = `""` 🔗

Optional name for the 3D physics layer 13. If left empty, the layer will display as "Layer 13".

String **layer_names/3d_physics/layer_14** = `""` 🔗

Optional name for the 3D physics layer 14. If left empty, the layer will display as "Layer 14".

String **layer_names/3d_physics/layer_15** = `""` 🔗

Optional name for the 3D physics layer 15. If left empty, the layer will display as "Layer 15".

String **layer_names/3d_physics/layer_16** = `""` 🔗

Optional name for the 3D physics layer 16. If left empty, the layer will display as "Layer 16".

String **layer_names/3d_physics/layer_17** = `""` 🔗

Optional name for the 3D physics layer 17. If left empty, the layer will display as "Layer 17".

String **layer_names/3d_physics/layer_18** = ""

Optional name for the 3D physics layer 18. If left empty, the layer will display as "Layer 18".

String **layer_names/3d_physics/layer_19** = ""

Optional name for the 3D physics layer 19. If left empty, the layer will display as "Layer 19".

String **layer_names/3d_physics/layer_20** = ""

Optional name for the 3D physics layer 20. If left empty, the layer will display as "Layer 20".

String **layer_names/3d_physics/layer_21** = ""

Optional name for the 3D physics layer 21. If left empty, the layer will display as "Layer 21".

String **layer_names/3d_physics/layer_22** = ""

Optional name for the 3D physics layer 22. If left empty, the layer will display as "Layer 22".

String **layer_names/3d_physics/layer_23** = ""

Optional name for the 3D physics layer 23. If left empty, the layer will display as "Layer 23".

String **layer_names/3d_physics/layer_24** = ""

Optional name for the 3D physics layer 24. If left empty, the layer will display as "Layer 24".

String **layer_names/3d_physics/layer_25** = ""

Optional name for the 3D physics layer 25. If left empty, the layer will display as "Layer 25".

String **layer_names/3d_physics/layer_26** = ""

Optional name for the 3D physics layer 26. If left empty, the layer will display as "Layer 26".

String **layer_names/3d_physics/layer_27** = ""

Optional name for the 3D physics layer 27. If left empty, the layer will display as "Layer 27".

String **layer_names/3d_physics/layer_28** = ""

Optional name for the 3D physics layer 28. If left empty, the layer will display as "Layer 28".

String **layer_names/3d_physics/layer_29** = ""

Optional name for the 3D physics layer 29. If left empty, the layer will display as "Layer 29".

String **layer_names/3d_physics/layer_30** = `""`

Optional name for the 3D physics layer 30. If left empty, the layer will display as "Layer 30".

String **layer_names/3d_physics/layer_31** = `""`

Optional name for the 3D physics layer 31. If left empty, the layer will display as "Layer 31".

String **layer_names/3d_physics/layer_32** = `""`

Optional name for the 3D physics layer 32. If left empty, the layer will display as "Layer 32".

String **layer_names/3d_render/layer_1** = `""`

Optional name for the 3D render layer 1. If left empty, the layer will display as "Layer 1".

String **layer_names/3d_render/layer_2** = `""`

Optional name for the 3D render layer 2. If left empty, the layer will display as "Layer 2".

String **layer_names/3d_render/layer_3** = `""`

Optional name for the 3D render layer 3. If left empty, the layer will display as "Layer 3".

String **layer_names/3d_render/layer_4** = `""`

Optional name for the 3D render layer 4. If left empty, the layer will display as "Layer 4".

String **layer_names/3d_render/layer_5** = `""`

Optional name for the 3D render layer 5. If left empty, the layer will display as "Layer 5".

String **layer_names/3d_render/layer_6** = `""`

Optional name for the 3D render layer 6. If left empty, the layer will display as "Layer 6".

String **layer_names/3d_render/layer_7** = `""`

Optional name for the 3D render layer 7. If left empty, the layer will display as "Layer 7".

String **layer_names/3d_render/layer_8** = `""`

Optional name for the 3D render layer 8. If left empty, the layer will display as "Layer 8".

String **layer_names/3d_render/layer_9** = `""`

Optional name for the 3D render layer 9. If left empty, the layer will display as "Layer 9".

String **layer_names/3d_render/layer_10** = `""`

Optional name for the 3D render layer 10. If left empty, the layer will display as "Layer 10".

String **layer_names/3d_render/layer_11** = `""`

Optional name for the 3D render layer 11. If left empty, the layer will display as "Layer 11".

String **layer_names/3d_render/layer_12** = `""`

Optional name for the 3D render layer 12. If left empty, the layer will display as "Layer 12".

String **layer_names/3d_render/layer_13** = `""`

Optional name for the 3D render layer 13. If left empty, the layer will display as "Layer 13".

String **layer_names/3d_render/layer_14** = `""`

Optional name for the 3D render layer 14. If left empty, the layer will display as "Layer 14".

String **layer_names/3d_render/layer_15** = `""`

Optional name for the 3D render layer 15. If left empty, the layer will display as "Layer 15".

String **layer_names/3d_render/layer_16** = `""`

Optional name for the 3D render layer 16. If left empty, the layer will display as "Layer 16".

String **layer_names/3d_render/layer_17** = `""`

Optional name for the 3D render layer 17. If left empty, the layer will display as "Layer 17".

String **layer_names/3d_render/layer_18** = `""`

Optional name for the 3D render layer 18. If left empty, the layer will display as "Layer 18".

String **layer_names/3d_render/layer_19** = `""`

Optional name for the 3D render layer 19. If left empty, the layer will display as "Layer 19".

String **layer_names/3d_render/layer_20** = `""`

Optional name for the 3D render layer 20. If left empty, the layer will display as "Layer 20".

String **layer_names/avoidance/layer_1** = `""`

Optional name for the navigation avoidance layer 1. If left empty, the layer will display as "Layer 1".

String **layer_names/avoidance/layer_2** = `""`

Optional name for the navigation avoidance layer 2. If left empty, the layer will display as "Layer 2".

String **layer_names/avoidance/layer_3** = `""`

Optional name for the navigation avoidance layer 3. If left empty, the layer will display as "Layer 3".

String **layer_names/avoidance/layer_4** = `""`

Optional name for the navigation avoidance layer 4. If left empty, the layer will display as "Layer 4".

String **layer_names/avoidance/layer_5** = `""`

Optional name for the navigation avoidance layer 5. If left empty, the layer will display as "Layer 5".

String **layer_names/avoidance/layer_6** = `""`

Optional name for the navigation avoidance layer 6. If left empty, the layer will display as "Layer 6".

String **layer_names/avoidance/layer_7** = `""`

Optional name for the navigation avoidance layer 7. If left empty, the layer will display as "Layer 7".

String **layer_names/avoidance/layer_8** = `""`

Optional name for the navigation avoidance layer 8. If left empty, the layer will display as "Layer 8".

String **layer_names/avoidance/layer_9** = `""`

Optional name for the navigation avoidance layer 9. If left empty, the layer will display as "Layer 9".

String **layer_names/avoidance/layer_10** = `""`

Optional name for the navigation avoidance layer 10. If left empty, the layer will display as "Layer 10".

String **layer_names/avoidance/layer_11** = `""`

Optional name for the navigation avoidance layer 11. If left empty, the layer will display as "Layer 11".

String **layer_names/avoidance/layer_12** = `""`

Optional name for the navigation avoidance layer 12. If left empty, the layer will display as "Layer 12".

String **layer_names/avoidance/layer_13** = `""`

Optional name for the navigation avoidance layer 13. If left empty, the layer will display as "Layer 13".

String **layer_names/avoidance/layer_14** = `""`

Optional name for the navigation avoidance layer 14. If left empty, the layer will display as "Layer 14".

String **layer_names/avoidance/layer_15** = `""`

Optional name for the navigation avoidance layer 15. If left empty, the layer will display as "Layer 15".

String **layer_names/avoidance/layer_16** = `""`

Optional name for the navigation avoidance layer 16. If left empty, the layer will display as "Layer 16".

String **layer_names/avoidance/layer_17** = `""`

Optional name for the navigation avoidance layer 17. If left empty, the layer will display as "Layer 17".

String **layer_names/avoidance/layer_18** = `""`

Optional name for the navigation avoidance layer 18. If left empty, the layer will display as "Layer 18".

String **layer_names/avoidance/layer_19** = `""`

Optional name for the navigation avoidance layer 19. If left empty, the layer will display as "Layer 19".

String **layer_names/avoidance/layer_20** = `""`

Optional name for the navigation avoidance layer 20. If left empty, the layer will display as "Layer 20".

String **layer_names/avoidance/layer_21** = `""`

Optional name for the navigation avoidance layer 21. If left empty, the layer will display as "Layer 21".

String **layer_names/avoidance/layer_22** = `""`

Optional name for the navigation avoidance layer 22. If left empty, the layer will display as "Layer 22".

String **layer_names/avoidance/layer_23** = `""`

Optional name for the navigation avoidance layer 23. If left empty, the layer will display as "Layer 23".

String **layer_names/avoidance/layer_24** = `""`

Optional name for the navigation avoidance layer 24. If left empty, the layer will display as "Layer 24".

String **layer_names/avoidance/layer_25** = `""`

Optional name for the navigation avoidance layer 25. If left empty, the layer will display as "Layer 25".

String **layer_names/avoidance/layer_26** = `""`

Optional name for the navigation avoidance layer 26. If left empty, the layer will display as "Layer 26".

String **layer_names/avoidance/layer_27** = `""`

Optional name for the navigation avoidance layer 27. If left empty, the layer will display as "Layer 27".

String **layer_names/avoidance/layer_28** = `""`

Optional name for the navigation avoidance layer 28. If left empty, the layer will display as "Layer 28".

String **layer_names/avoidance/layer_29** = `""`

Optional name for the navigation avoidance layer 29. If left empty, the layer will display as "Layer 29".

String **layer_names/avoidance/layer_30** = `""`

Optional name for the navigation avoidance layer 30. If left empty, the layer will display as "Layer 30".

String **layer_names/avoidance/layer_31** = `""`

Optional name for the navigation avoidance layer 31. If left empty, the layer will display as "Layer 31".

String **layer_names/avoidance/layer_32** = `""`

Optional name for the navigation avoidance layer 32. If left empty, the layer will display as "Layer 32".

int **memory/limits/message_queue/max_size_mb** = `32`

Godot uses a message queue to defer some function calls. If you run out of space on it (you will see an error), you can increase the size here.

float **navigation/2d/default_cell_size** = `1.0`

Default cell size for 2D navigation maps. See NavigationServer2D.map_set_cell_size.

float **navigation/2d/default_edge_connection_margin** = `1.0`

Default edge connection margin for 2D navigation maps. See NavigationServer2D.map_set_edge_connection_margin.

float **navigation/2d/default_link_connection_radius** = `4.0`

Default link connection radius for 2D navigation maps. See NavigationServer2D.map_set_link_connection_radius.

bool **navigation/2d/use_edge_connections** = `true`

If enabled 2D navigation regions will use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin. This setting only affects World2D default navigation maps.

float **navigation/3d/default_cell_height** = `0.25`

Default cell height for 3D navigation maps. See NavigationServer3D.map_set_cell_height.

float **navigation/3d/default_cell_size** = `0.25`

Default cell size for 3D navigation maps. See NavigationServer3D.map_set_cell_size.

float **navigation/3d/default_edge_connection_margin** = `0.25`

Default edge connection margin for 3D navigation maps. See NavigationServer3D.map_set_edge_connection_margin.

float **navigation/3d/default_link_connection_radius** = `1.0`

Default link connection radius for 3D navigation maps. See NavigationServer3D.map_set_link_connection_radius.

Vector3 **navigation/3d/default_up** = `Vector3(0, 1, 0)`

Default up orientation for 3D navigation maps. See NavigationServer3D.map_set_up.

float **navigation/3d/merge_rasterizer_cell_scale** = `1.0`

Default merge rasterizer cell scale for 3D navigation maps. See NavigationServer3D.map_set_merge_rasterizer_cell_scale.

bool **navigation/3d/use_edge_connections** = `true`

If enabled 3D navigation regions will use edge connections to connect with other navigation regions within proximity of the navigation map edge connection margin. This setting only affects World3D default navigation maps.

---

bool **navigation/avoidance/thread_model/avoidance_use_high_priority_threads** = `true`

If enabled and avoidance calculations use multiple threads the threads run with high priority.

---

bool **navigation/avoidance/thread_model/avoidance_use_multiple_threads** = `true`

If enabled the avoidance calculations use multiple threads.

---

bool **navigation/baking/thread_model/baking_use_high_priority_threads** = `true`

If enabled and async navmesh baking uses multiple threads the threads run with high priority.

---

bool **navigation/baking/thread_model/baking_use_multiple_threads** = `true`

If enabled the async navmesh baking uses multiple threads.

---

bool **navigation/baking/use_crash_prevention_checks** = `true`

If enabled, and baking would potentially lead to an engine crash, the baking will be interrupted and an error message with explanation will be raised.

---

int **network/limits/debugger/max_chars_per_second** = `32768`

Maximum number of characters allowed to send as output from the debugger. Over this value, content is dropped. This helps not to stall the debugger connection.

---

int **network/limits/debugger/max_errors_per_second** = `400`

Maximum number of errors allowed to be sent from the debugger. Over this value, content is dropped. This helps not to stall the debugger connection.

---

int **network/limits/debugger/max_queued_messages** = `2048`

Maximum number of messages in the debugger queue. Over this value, content is dropped. This helps to limit the debugger memory usage.

---

int **network/limits/debugger/max_warnings_per_second** = `400`

Maximum number of warnings allowed to be sent from the debugger. Over this value, content is dropped. This helps not to stall the debugger connection.

---

int **network/limits/packet_peer_stream/max_buffer_po2** = `16`

Default size of packet peer stream for deserializing Godot data (in bytes, specified as a power of two). The default value `16` is equal to 65,536 bytes. Over this size, data is dropped.

---

int **network/limits/tcp/connect_timeout_seconds** = `30`

Timeout (in seconds) for connection attempts using TCP.

---

int **network/limits/webrtc/max_channel_in_buffer_kb** = `64`

Maximum size (in kiB) for the WebRTCDataChannel input buffer.

String **network/tls/certificate_bundle_override** = `""`

The CA certificates bundle to use for TLS connections. If this is set to a non-empty value, this will *override* Godot's default Mozilla certificate bundle [https://github.com/godotengine/godot/blob/master/thirdparty/certs/ca-certificates.crt]. If left empty, the default certificate bundle will be used.

If in doubt, leave this setting empty.

---

float **physics/2d/default_angular_damp** = `1.0`

The default rotational motion damping in 2D. Damping is used to gradually slow down physical objects over time. RigidBodies will fall back to this value when combining their own damping values and no area damping value is present.

Suggested values are in the range `0` to `30`. At value `0` objects will keep moving with the same velocity. Greater values will stop the object faster. A value equal to or greater than the physics tick rate (physics/common/physics_ticks_per_second) will bring the object to a stop in one iteration.

Note: Godot damping calculations are velocity-dependent, meaning bodies moving faster will take a longer time to come to rest. They do not simulate inertia, friction, or air resistance. Therefore heavier or larger bodies will lose speed at the same proportional rate as lighter or smaller bodies.

During each physics tick, Godot will multiply the linear velocity of RigidBodies by `1.0 - combined_damp / physics_ticks_per_second`. By default, bodies combine damp factors: `combined_damp` is the sum of the damp value of the body and this value or the area's value the body is in. See DampMode.

Warning: Godot's damping calculations are simulation tick rate dependent. Changing physics/common/physics_ticks_per_second may significantly change the outcomes and feel of your simulation. This is true for the entire range of damping values greater than 0. To get back to a similar feel, you also need to change your damp values. This needed change is not proportional and differs from case to case.

---

float **physics/2d/default_gravity** = `980.0`

The default gravity strength in 2D (in pixels per second squared).

Note: This property is only read when the project starts. To change the default gravity at runtime, use the following code sample:

GDScript

```
# Set the default gravity strength to 980.
PhysicsServer2D.area_set_param(get_viewport().find_world_2d().space,
PhysicsServer2D.AREA_PARAM_GRAVITY, 980)
```

C#

```
// Set the default gravity strength to 980.
PhysicsServer2D.AreaSetParam(GetViewport().FindWorld2D().Space,
PhysicsServer2D.AreaParameter.Gravity, 980);
```

Vector2 **physics/2d/default_gravity_vector** = `Vector2(0, 1)`

The default gravity direction in 2D.

Note: This property is only read when the project starts. To change the default gravity vector at runtime, use the following code sample:

GDScript

```
# Set the default gravity direction to `Vector2(0, 1)`.
PhysicsServer2D.area_set_param(get_viewport().find_world_2d().space,
PhysicsServer2D.AREA_PARAM_GRAVITY_VECTOR, Vector2.DOWN)
```

C#

```
// Set the default gravity direction to `Vector2(0, 1)`.
PhysicsServer2D.AreaSetParam(GetViewport().FindWorld2D().Space,
PhysicsServer2D.AreaParameter.GravityVector, Vector2.Down)
```

---

float **physics/2d/default_linear_damp** = `0.1`

The default linear motion damping in 2D. Damping is used to gradually slow down physical objects over time. RigidBodies will fall back to this value when combining their own damping values and no area damping value is present.

Suggested values are in the range `0` to `30`. At value `0` objects will keep moving with the same velocity. Greater values will stop the object faster. A value equal to or greater than the physics tick rate (physics/common/physics_ticks_per_second) will bring the object to a stop in one iteration.

Note: Godot damping calculations are velocity-dependent, meaning bodies moving faster will take a longer time to come to rest. They do not simulate inertia, friction, or air resistance. Therefore heavier or larger bodies will lose speed at the same proportional rate as lighter or smaller bodies.

During each physics tick, Godot will multiply the linear velocity of RigidBodies by `1.0 - combined_damp / physics_ticks_per_second`, where `combined_damp` is the sum of the linear damp of the body and this value, or the area's value the body is in, assuming the body defaults to combine damp values. See DampMode.

Warning: Godot's damping calculations are simulation tick rate dependent. Changing physics/common/physics_ticks_per_second may significantly change the outcomes and feel of your simulation. This is true for the entire range of damping values greater than 0. To get back to a similar feel, you also need to change your damp values. This needed change is not proportional and differs from case to case.

String **physics/2d/physics_engine** = `"DEFAULT"`

Sets which physics engine to use for 2D physics.

"DEFAULT" and "GodotPhysics2D" are the same, as there is currently no alternative 2D physics server implemented.

bool **physics/2d/run_on_separate_thread** = `false`

If `true`, the 2D physics server runs on a separate thread, making better use of multi-core CPUs. If `false`, the 2D physics server runs on the main thread. Running the physics server on a separate thread can increase performance, but restricts API access to only physics process.

float **physics/2d/sleep_threshold_angular** = `0.139626`

Threshold angular velocity under which a 2D physics body will be considered inactive. See PhysicsServer2D.SPACE_PARAM_BODY_ANGULAR_VELOCITY_SLEEP_THRESHOLD.

float **physics/2d/sleep_threshold_linear** = `2.0`

Threshold linear velocity under which a 2D physics body will be considered inactive. See PhysicsServer2D.SPACE_PARAM_BODY_LINEAR_VELOCITY_SLEEP_THRESHOLD.

float **physics/2d/solver/contact_max_allowed_penetration** = `0.3`

Maximum distance a shape can penetrate another shape before it is considered a collision. See PhysicsServer2D.SPACE_PARAM_CONTACT_MAX_ALLOWED_PENETRATION.

float **physics/2d/solver/contact_max_separation** = `1.5`

Maximum distance a shape can be from another before they are considered separated and the contact is discarded. See PhysicsServer2D.SPACE_PARAM_CONTACT_MAX_SEPARATION.

float **physics/2d/solver/contact_recycle_radius** = `1.0`

Maximum distance a pair of bodies has to move before their collision status has to be recalculated. See PhysicsServer2D.SPACE_PARAM_CONTACT_RECYCLE_RADIUS.

float **physics/2d/solver/default_constraint_bias** = `0.2`

Default solver bias for all physics constraints. Defines how much bodies react to enforce constraints. See PhysicsServer2D.SPACE_PARAM_CONSTRAINT_DEFAULT_BIAS.

Individual constraints can have a specific bias value (see Joint2D.bias).

float **physics/2d/solver/default_contact_bias** = `0.8`

Default solver bias for all physics contacts. Defines how much bodies react to enforce contact separation. See PhysicsServer2D.SPACE_PARAM_CONTACT_DEFAULT_BIAS.

Individual shapes can have a specific bias value (see Shape2D.custom_solver_bias).

---

int **physics/2d/solver/solver_iterations** = `16`

Number of solver iterations for all contacts and constraints. The greater the number of iterations, the more accurate the collisions will be. However, a greater number of iterations requires more CPU power, which can decrease performance. See PhysicsServer2D.SPACE_PARAM_SOLVER_ITERATIONS.

---

float **physics/2d/time_before_sleep** = `0.5`

Time (in seconds) of inactivity before which a 2D physics body will put to sleep. See PhysicsServer2D.SPACE_PARAM_BODY_TIME_TO_SLEEP.

---

float **physics/3d/default_angular_damp** = `0.1`

The default rotational motion damping in 3D. Damping is used to gradually slow down physical objects over time. RigidBodies will fall back to this value when combining their own damping values and no area damping value is present.

Suggested values are in the range `0` to `30`. At value `0` objects will keep moving with the same velocity. Greater values will stop the object faster. A value equal to or greater than the physics tick rate (physics/common/physics_ticks_per_second) will bring the object to a stop in one iteration.

Note: Godot damping calculations are velocity-dependent, meaning bodies moving faster will take a longer time to come to rest. They do not simulate inertia, friction, or air resistance. Therefore heavier or larger bodies will lose speed at the same proportional rate as lighter or smaller bodies.

During each physics tick, Godot will multiply the angular velocity of RigidBodies by `1.0 - combined_damp / physics_ticks_per_second`. By default, bodies combine damp factors: `combined_damp` is the sum of the damp value of the body and this value or the area's value the body is in. See DampMode.

Warning: Godot's damping calculations are simulation tick rate dependent. Changing physics/common/physics_ticks_per_second may significantly change the outcomes and feel of your simulation. This is true for the entire range of damping values greater than 0. To get back to a similar feel, you also need to change your damp values. This needed change is not proportional and differs from case to case.

---

float **physics/3d/default_gravity** = `9.8`

The default gravity strength in 3D (in meters per second squared).

Note: This property is only read when the project starts. To change the default gravity at runtime, use the following code sample:

GDScript

```
# Set the default gravity strength to 9.8.
PhysicsServer3D.area_set_param(get_viewport().find_world_3d().space,
PhysicsServer3D.AREA_PARAM_GRAVITY, 9.8)
```

C#

```
// Set the default gravity strength to 9.8.
PhysicsServer3D.AreaSetParam(GetViewport().FindWorld3D().Space,
PhysicsServer3D.AreaParameter.Gravity, 9.8);
```

Vector3 **physics/3d/default_gravity_vector** = `Vector3(0, -1, 0)`

The default gravity direction in 3D.

Note: This property is only read when the project starts. To change the default gravity vector at runtime, use the following code sample:

GDScript

```
# Set the default gravity direction to `Vector3(0, -1, 0)`.
PhysicsServer3D.area_set_param(get_viewport().find_world_3d().space,
PhysicsServer3D.AREA_PARAM_GRAVITY_VECTOR, Vector3.DOWN)
```

C#

```
// Set the default gravity direction to `Vector3(0, -1, 0)`.
PhysicsServer3D.AreaSetParam(GetViewport().FindWorld3D().Space,
PhysicsServer3D.AreaParameter.GravityVector, Vector3.Down)
```

---

float **physics/3d/default_linear_damp** = `0.1`

The default linear motion damping in 3D. Damping is used to gradually slow down physical objects over time. RigidBodies will fall back to this value when combining their own damping values and no area damping value is present.

Suggested values are in the range `0` to `30`. At value `0` objects will keep moving with the same velocity. Greater values will stop the object faster. A value equal to or greater than the physics tick rate (physics/common/physics_ticks_per_second) will bring the object to a stop in one iteration.

Note: Godot damping calculations are velocity-dependent, meaning bodies moving faster will take a longer time to come to rest. They do not simulate inertia, friction, or air resistance. Therefore heavier or larger bodies will lose speed at the same proportional rate as lighter or smaller bodies.

During each physics tick, Godot will multiply the linear velocity of RigidBodies by `1.0 - combined_damp / physics_ticks_per_second`. By default, bodies combine damp factors: `combined_damp` is the sum of the damp value of the body and this value or the area's value the body is in. See DampMode.

Warning: Godot's damping calculations are simulation tick rate dependent. Changing physics/common/physics_ticks_per_second may significantly change the outcomes and feel of your simulation. This is true for the entire range of damping values greater than 0. To get back to a similar feel, you also need to change your damp values. This needed change is not proportional and differs from case to case.

String **physics/3d/physics_engine** = `"DEFAULT"`

Sets which physics engine to use for 3D physics.

"DEFAULT" and "GodotPhysics3D" are the same, as there is currently no alternative 3D physics server implemented.

---

bool **physics/3d/run_on_separate_thread** = `false`

If `true`, the 3D physics server runs on a separate thread, making better use of multi-core CPUs. If `false`, the 3D physics server runs on the main thread. Running the physics server on a separate thread can increase performance, but restricts API access to only physics process.

---

float **physics/3d/sleep_threshold_angular** = `0.139626`

Threshold angular velocity under which a 3D physics body will be considered inactive. See PhysicsServer3D.SPACE_PARAM_BODY_ANGULAR_VELOCITY_SLEEP_THRESHOLD.

---

float **physics/3d/sleep_threshold_linear** = `0.1`

Threshold linear velocity under which a 3D physics body will be considered inactive. See PhysicsServer3D.SPACE_PARAM_BODY_LINEAR_VELOCITY_SLEEP_THRESHOLD.

---

float **physics/3d/solver/contact_max_allowed_penetration** = `0.01`

Maximum distance a shape can penetrate another shape before it is considered a collision. See PhysicsServer3D.SPACE_PARAM_CONTACT_MAX_ALLOWED_PENETRATION.

---

float **physics/3d/solver/contact_max_separation** = `0.05`

Maximum distance a shape can be from another before they are considered separated and the contact is discarded. See PhysicsServer3D.SPACE_PARAM_CONTACT_MAX_SEPARATION.

---

float **physics/3d/solver/contact_recycle_radius** = `0.01`

Maximum distance a pair of bodies has to move before their collision status has to be recalculated. See PhysicsServer3D.SPACE_PARAM_CONTACT_RECYCLE_RADIUS.

---

float **physics/3d/solver/default_contact_bias** = `0.8`

Default solver bias for all physics contacts. Defines how much bodies react to enforce contact separation. See PhysicsServer3D.SPACE_PARAM_CONTACT_DEFAULT_BIAS.

Individual shapes can have a specific bias value (see Shape3D.custom_solver_bias).

---

int **physics/3d/solver/solver_iterations** = `16`

Number of solver iterations for all contacts and constraints. The greater the number of iterations, the more accurate the collisions will be. However, a greater number of iterations requires more CPU power, which can decrease performance. See PhysicsServer3D.SPACE_PARAM_SOLVER_ITERATIONS.

---

float **physics/3d/time_before_sleep** = `0.5`

Time (in seconds) of inactivity before which a 3D physics body will put to sleep. See PhysicsServer3D.SPACE_PARAM_BODY_TIME_TO_SLEEP.

---

bool **physics/common/enable_object_picking** = `true`

Enables Viewport.physics_object_picking on the root viewport.

---

int **physics/common/max_physics_steps_per_frame** = `8`

Controls the maximum number of physics steps that can be simulated each rendered frame. The default value is tuned to avoid "spiral of death" situations where expensive physics simulations trigger more expensive simulations indefinitely. However, the game will appear to slow down if the rendering FPS is less than `1 / max_physics_steps_per_frame` of physics/common/physics_ticks_per_second. This occurs even if `delta` is consistently used in physics calculations. To avoid this, increase physics/common/max_physics_steps_per_frame if you have increased physics/common/physics_ticks_per_second significantly above its default value.

Note: This property is only read when the project starts. To change the maximum number of simulated physics steps per frame at runtime, set Engine.max_physics_steps_per_frame instead.

---

bool **physics/common/physics_interpolation** = `false`

If `true`, the renderer will interpolate the transforms of physics objects between the last two transforms, so that smooth motion is seen even when physics ticks do not coincide with rendered frames. See also Node.physics_interpolation_mode and Node.reset_physics_interpolation.

Note: If `true`, the physics jitter fix should be disabled by setting physics/common/physics_jitter_fix to `0.0`.

Note: This property is only read when the project starts. To toggle physics interpolation at runtime, set SceneTree.physics_interpolation instead.

Note: This feature is currently only implemented in the 2D renderer.

---

float **physics/common/physics_jitter_fix** = `0.5`

Controls how much physics ticks are synchronized with real time. For 0 or less, the ticks are synchronized. Such values are recommended for network games, where clock synchronization matters. Higher values cause higher deviation of in-game clock and real clock, but allows smoothing out framerate jitters. The default value of 0.5 should be good enough for most; values above 2 could cause the game to react to dropped frames with a noticeable delay and are not recommended.

Note: When using a physics interpolation solution (such as enabling physics/common/physics_interpolation or using a custom solution), the physics jitter fix should be disabled by setting physics/common/physics_jitter_fix to `0.0`.

Note: This property is only read when the project starts. To change the physics jitter fix at runtime, set Engine.physics_jitter_fix instead.

---

int **physics/common/physics_ticks_per_second** = `60`

The number of fixed iterations per second. This controls how often physics simulation and Node._physics_process methods are run. See also application/run/max_fps.

Note: This property is only read when the project starts. To change the physics FPS at runtime, set Engine.physics_ticks_per_second instead.

Note: Only physics/common/max_physics_steps_per_frame physics ticks may be simulated per rendered frame at most. If more physics ticks have to be simulated per rendered frame to keep up with rendering, the project will appear to slow down (even if `delta` is used consistently in physics calculations). Therefore, it is recommended to also increase physics/common/max_physics_steps_per_frame if increasing physics/common/physics_ticks_per_second significantly above its default value.

---

int **rendering/2d/sdf/oversize** = `1`

Controls how much of the original viewport size should be covered by the 2D signed distance field. This SDF can be sampled in CanvasItem shaders and is used for GPUParticles2D collision. Higher values allow portions of occluders located outside the viewport to still be taken into account in the generated signed distance field, at the cost of performance. If you notice particles falling through LightOccluder2Ds as the occluders leave the viewport, increase this setting.

The percentage specified is added on each axis and on both sides. For example, with the default setting of 120%, the signed distance field will cover 20% of the viewport's size outside the viewport on each side (top, right, bottom, left).

Note: This property is only read when the project starts. To change the 2D SDF oversizing percentage at runtime, use RenderingServer.viewport_set_sdf_oversize_and_scale instead.

---

int **rendering/2d/sdf/scale** = `1`

The resolution scale to use for the 2D signed distance field. Higher values lead to a more precise and more stable signed distance field as the camera moves, at the cost of performance. The default value (50%) renders at half the resolution of the viewport size on each axis, which means the SDF is generated with 25% of the viewport's pixel count.

Note: This property is only read when the project starts. To change the 2D SDF resolution scale at runtime, use RenderingServer.viewport_set_sdf_oversize_and_scale instead.

---

int **rendering/2d/shadow_atlas/size** = `2048`

The size of the 2D shadow atlas in pixels. Higher values result in more precise Light2D shadows, at the cost of performance and video memory usage. The specified value is rounded up to the nearest power of 2.

Note: This property is only read when the project starts. To change the 2D shadow atlas size at runtime, use RenderingServer.canvas_set_shadow_texture_size instead.

---

bool **rendering/2d/snap/snap_2d_transforms_to_pixel** = `false` 🔗

If `true`, CanvasItem nodes will internally snap to full pixels. Useful for low-resolution pixel art games. Their position can still be sub-pixel, but the decimals will not have effect as the position is rounded. This can lead to a crisper appearance at the cost of less smooth movement, especially when Camera2D smoothing is enabled.

Note: This property is only read when the project starts. To toggle 2D transform snapping at runtime, use RenderingServer.viewport_set_snap_2d_transforms_to_pixel on the root Viewport instead.

Note: Control nodes are snapped to the nearest pixel by default. This is controlled by gui/common/snap_controls_to_pixels.

Note: It is not recommended to use this setting together with rendering/2d/snap/snap_2d_vertices_to_pixel, as movement may appear even less smooth. Prefer only enabling this setting instead.

---

bool **rendering/2d/snap/snap_2d_vertices_to_pixel** = `false` 🔗

If `true`, vertices of CanvasItem nodes will snap to full pixels. Useful for low-resolution pixel art games. Only affects the final vertex positions, not the transforms. This can lead to a crisper appearance at the cost of less smooth movement, especially when Camera2D smoothing is enabled.

Note: This property is only read when the project starts. To toggle 2D vertex snapping at runtime, use RenderingServer.viewport_set_snap_2d_vertices_to_pixel on the root Viewport instead.

Note: Control nodes are snapped to the nearest pixel by default. This is controlled by gui/common/snap_controls_to_pixels.

Note: It is not recommended to use this setting together with rendering/2d/snap/snap_2d_transforms_to_pixel, as movement may appear even less smooth. Prefer only enabling that setting instead.

---

int **rendering/anti_aliasing/quality/msaa_2d** = `0` 🔗

Sets the number of MSAA samples to use for 2D/Canvas rendering (as a power of two). MSAA is used to reduce aliasing around the edges of polygons. A higher MSAA value results in smoother edges but can be significantly slower on some hardware, especially integrated graphics due to their limited memory bandwidth. This has no effect on shader-induced aliasing or texture aliasing.

Note: MSAA is only supported in the Forward+ and Mobile rendering methods, not Compatibility.

---

int **rendering/anti_aliasing/quality/msaa_3d** = `0` 🔗

Sets the number of MSAA samples to use for 3D rendering (as a power of two). MSAA is used to reduce aliasing around the edges of polygons. A higher MSAA value results in smoother edges but can be significantly slower on some hardware, especially integrated graphics due to their limited memory bandwidth. See also rendering/scaling_3d/mode for supersampling, which provides higher quality but is much more expensive. This has no effect on shader-induced aliasing or texture aliasing.

---

int **rendering/anti_aliasing/quality/screen_space_aa** = `0`

Sets the screen-space antialiasing mode for the default screen Viewport. Screen-space antialiasing works by selectively blurring edges in a post-process shader. It differs from MSAA which takes multiple coverage samples while rendering objects. Screen-space AA methods are typically faster than MSAA and will smooth out specular aliasing, but tend to make scenes appear blurry. The blurriness is partially counteracted by automatically using a negative mipmap LOD bias (see rendering/textures/default_filters/texture_mipmap_bias).

Another way to combat specular aliasing is to enable rendering/anti_aliasing/screen_space_roughness_limiter/enabled.

Note: Screen-space antialiasing is only supported in the Forward+ and Mobile rendering methods, not Compatibility.

bool **rendering/anti_aliasing/quality/use_debanding** = `false`

If `true`, uses a fast post-processing filter to make banding significantly less visible in 3D. 2D rendering is *not* affected by debanding unless the Environment.background_mode is Environment.BG_CANVAS.

In some cases, debanding may introduce a slightly noticeable dithering pattern. It's recommended to enable debanding only when actually needed since the dithering pattern will make lossless-compressed screenshots larger.

Note: This property is only read when the project starts. To set debanding at run-time, set Viewport.use_debanding on the root Viewport instead.

bool **rendering/anti_aliasing/quality/use_taa** = `false`

Enables Temporal Anti-Aliasing for the default screen Viewport. TAA works by jittering the camera and accumulating the images of the last rendered frames, motion vector rendering is used to account for camera and object motion. Enabling TAA can make the image blurrier, which is partially counteracted by automatically using a negative mipmap LOD bias (see rendering/textures/default_filters/texture_mipmap_bias).

Note: The implementation is not complete yet. Some visual instances such as particles and skinned meshes may show ghosting artifacts in motion.

Note: TAA is only supported in the Forward+ rendering method, not Mobile or Compatibility.

float **rendering/anti_aliasing/screen_space_roughness_limiter/amount** = `0.25`

Note: This property is only read when the project starts. To control the screen-space roughness limiter at runtime, call RenderingServer.screen_space_roughness_limiter_set_active instead.

bool **rendering/anti_aliasing/screen_space_roughness_limiter/enabled** = `true`

If `true`, enables a spatial filter to limit roughness in areas with high-frequency detail. This can help reduce specular aliasing to an extent, though not as much as enabling rendering/anti_aliasing/quality/use_taa. This filter has a small performance cost, so consider disabling it if it doesn't benefit your scene noticeably.

Note: The screen-space roughness limiter is only supported in the Forward+ and Mobile rendering methods, not Compatibility.

Note: This property is only read when the project starts. To control the screen-space roughness limiter at runtime, call RenderingServer.screen_space_roughness_limiter_set_active instead.

---

float **rendering/anti_aliasing/screen_space_roughness_limiter/limit** = `0.18` 🔗

Note: This property is only read when the project starts. To control the screen-space roughness limiter at runtime, call RenderingServer.screen_space_roughness_limiter_set_active instead.

---

int **rendering/camera/depth_of_field/depth_of_field_bokeh_quality** = `1` 🔗

Sets the quality of the depth of field effect. Higher quality takes more samples, which is slower but looks smoother.

---

int **rendering/camera/depth_of_field/depth_of_field_bokeh_shape** = `1` 🔗

Sets the depth of field shape. Can be Box, Hexagon, or Circle. Box is the fastest. Circle is the most realistic, but also the most expensive to compute.

---

bool **rendering/camera/depth_of_field/depth_of_field_use_jitter** = `false` 🔗

If `true`, jitters DOF samples to make effect slightly blurrier and hide lines created from low sample rates. This can result in a slightly grainy appearance when used with a low number of samples.

String **rendering/driver/depth_prepass/disable_for_vendors** = `"PowerVR,Mali,Adreno,Apple"` 🔗

Disables rendering/driver/depth_prepass/enable conditionally for certain vendors. By default, disables the depth prepass for mobile devices as mobile devices do not benefit from the depth prepass due to their unique architecture.

---

bool **rendering/driver/depth_prepass/enable** = `true` 🔗

If `true`, performs a previous depth pass before rendering 3D materials. This increases performance significantly in scenes with high overdraw, when complex materials and lighting are used. However, in scenes with few occluded surfaces, the depth prepass may reduce performance. If your game is viewed from a fixed angle that makes it easy to avoid overdraw (such as top-down or side-scrolling perspective), consider disabling the depth prepass to improve performance. This setting can be changed at run-time to optimize performance depending on the scene currently being viewed.

Note: Depth prepass is only supported when using the Forward+ or Compatibility rendering method. When using the Mobile rendering method, there is no depth prepass performed.

---

int **rendering/driver/threads/thread_model** = `1` 🔗

Experimental: This setting has several known bugs which can lead to crashing, especially when using particles or resizing the window. Not recommended for use in production at this stage.

The thread model to use for rendering. Rendering on a thread may improve performance, but synchronizing to the main thread can cause a bit more jitter.

Color **rendering/environment/defaults/default_clear_color** = `Color(0.3, 0.3, 0.3, 1)`

Default background clear color. Overridable per Viewport using its Environment. See Environment.background_mode and Environment.background_color in particular. To change this default color programmatically, use RenderingServer.set_default_clear_color.

String **rendering/environment/defaults/default_environment** = `""`

Environment that will be used as a fallback environment in case a scene does not specify its own environment. The default environment is loaded in at scene load time regardless of whether you have set an environment or not. If you do not rely on the fallback environment, you do not need to set this property.

---

int **rendering/environment/glow/upscale_mode** = `1`

Sets how the glow effect is upscaled before being copied onto the screen. Linear is faster, but looks blocky. Bicubic is slower but looks smooth.

---

int **rendering/environment/glow/upscale_mode.mobile** = `0`

Lower-end override for rendering/environment/glow/upscale_mode on mobile devices, due to performance concerns or driver support.

---

int **rendering/environment/screen_space_reflection/roughness_quality** = `1`

Sets the quality for rough screen-space reflections. Turning off will make all screen space reflections sharp, while higher values make rough reflections look better.

---

float **rendering/environment/ssao/adaptive_target** = `0.5`

Quality target to use when rendering/environment/ssao/quality is set to `Ultra`. A value of `0.0` provides a quality and speed similar to `Medium` while a value of `1.0` provides much higher quality than any of the other settings at the cost of performance.

---

int **rendering/environment/ssao/blur_passes** = `2`

Number of blur passes to use when computing screen-space ambient occlusion. A higher number will result in a smoother look, but will be slower to compute and will have less high-frequency detail.

---

float **rendering/environment/ssao/fadeout_from** = `50.0`

Distance at which the screen-space ambient occlusion effect starts to fade out. Use this hide ambient occlusion at great distances.

---

float **rendering/environment/ssao/fadeout_to** = `300.0`

Distance at which the screen-space ambient occlusion is fully faded out. Use this hide ambient occlusion at great distances.

---

bool **rendering/environment/ssao/half_size** = `true`

If `true`, screen-space ambient occlusion will be rendered at half size and then upscaled before being added to the scene. This is significantly faster but may miss small details. If `false`, screen-space ambient occlusion will be rendered at full size.

---

int **rendering/environment/ssao/quality** = `2`

Sets the quality of the screen-space ambient occlusion effect. Higher values take more samples and so will result in better quality, at the cost of performance. Setting to `Ultra` will use the rendering/environment/ssao/adaptive_target setting.

---

float **rendering/environment/ssil/adaptive_target** = `0.5`

Quality target to use when rendering/environment/ssil/quality is set to `Ultra`. A value of `0.0` provides a quality and speed similar to `Medium` while a value of `1.0` provides much higher quality than any of the other settings at the cost of performance. When using the adaptive target, the performance cost scales with the complexity of the scene.

---

int **rendering/environment/ssil/blur_passes** = `4`

Number of blur passes to use when computing screen-space indirect lighting. A higher number will result in a smoother look, but will be slower to compute and will have less high-frequency detail.

---

float **rendering/environment/ssil/fadeout_from** = `50.0`

Distance at which the screen-space indirect lighting effect starts to fade out. Use this hide screen-space indirect lighting at great distances.

---

float **rendering/environment/ssil/fadeout_to** = `300.0`

Distance at which the screen-space indirect lighting is fully faded out. Use this hide screen-space indirect lighting at great distances.

---

bool **rendering/environment/ssil/half_size** = `true`

If `true`, screen-space indirect lighting will be rendered at half size and then upscaled before being added to the scene. This is significantly faster but may miss small details and may result in some objects appearing to glow at their edges.

---

int **rendering/environment/ssil/quality** = `2`

Sets the quality of the screen-space indirect lighting effect. Higher values take more samples and so will result in better quality, at the cost of performance. Setting to `Ultra` will use the rendering/environment/ssil/adaptive_target setting.

---

float **rendering/environment/subsurface_scattering/subsurface_scattering_depth_scale** = `0.01`

Scales the depth over which the subsurface scattering effect is applied. A high value may allow light to scatter into a part of the mesh or another mesh that is close in screen space but far in depth. See also rendering/environment/subsurface_scattering/subsurface_scattering_scale.

Note: This property is only read when the project starts. To set the subsurface scattering depth scale at runtime, call RenderingServer.sub_surface_scattering_set_scale instead.

---

int **rendering/environment/subsurface_scattering/subsurface_scattering_quality** = `1`

Sets the quality of the subsurface scattering effect. Higher values are slower but look nicer. This affects the rendering of materials that have BaseMaterial3D.subsurf_scatter_enabled set to `true`, along with ShaderMaterials that set `SSS_STRENGTH`.

Note: This property is only read when the project starts. To set the subsurface scattering quality at runtime, call RenderingServer.sub_surface_scattering_set_quality instead.

---

float **rendering/environment/subsurface_scattering/subsurface_scattering_scale** = `0.05`

Scales the distance over which samples are taken for subsurface scattering effect. Changing this does not impact performance, but higher values will result in significant artifacts as the samples will become obviously spread out. A lower value results in a smaller spread of scattered light. See also rendering/environment/subsurface_scattering/subsurface_scattering_depth_scale.

Note: This property is only read when the project starts. To set the subsurface scattering scale at runtime, call RenderingServer.sub_surface_scattering_set_scale instead.

---

int **rendering/environment/volumetric_fog/use_filter** = `1`

Enables filtering of the volumetric fog effect prior to integration. This substantially blurs the fog which reduces fine details but also smooths out harsh edges and aliasing artifacts. Disable when more detail is required.

---

int **rendering/environment/volumetric_fog/volume_depth** = `64`

Number of slices to use along the depth of the froxel buffer for volumetric fog. A lower number will be more efficient but may result in artifacts appearing during camera movement. See also Environment.volumetric_fog_length.

---

int **rendering/environment/volumetric_fog/volume_size** = `64`

Base size used to determine size of froxel buffer in the camera X-axis and Y-axis. The final size is scaled by the aspect ratio of the screen, so actual values may differ from what is set. Set a larger size for more detailed fog, set a smaller size for better performance.

String **rendering/gl_compatibility/driver**

Sets the driver to be used by the renderer when using the Compatibility renderer. This property can not be edited directly, instead, set the driver using the platform-specific overrides.

String **rendering/gl_compatibility/driver.android**

Android override for rendering/gl_compatibility/driver.

String **rendering/gl_compatibility/driver.ios**

iOS override for rendering/gl_compatibility/driver.

String **rendering/gl_compatibility/driver.linuxbsd**

LinuxBSD override for rendering/gl_compatibility/driver.

String **rendering/gl_compatibility/driver.macos**

macOS override for rendering/gl_compatibility/driver.

String **rendering/gl_compatibility/driver.web**

Web override for rendering/gl_compatibility/driver.

String **rendering/gl_compatibility/driver.windows**

Windows override for rendering/gl_compatibility/driver.

---

bool **rendering/gl_compatibility/fallback_to_angle** = `true`

If `true`, the compatibility renderer will fall back to ANGLE if native OpenGL is not supported or the device is listed in rendering/gl_compatibility/force_angle_on_devices.

Note: This setting is implemented only on Windows.

---

bool **rendering/gl_compatibility/fallback_to_gles** = `true`

If `true`, the compatibility renderer will fall back to OpenGLES if desktop OpenGL is not supported.

Note: This setting is implemented only on Linux/X11.

---

bool **rendering/gl_compatibility/fallback_to_native** = `true`

If `true`, the compatibility renderer will fall back to native OpenGL if ANGLE over Metal is not supported.

Note: This setting is implemented only on macOS.

Array **rendering/gl_compatibility/force_angle_on_devices**

An Array of devices which should always use the ANGLE renderer.

Each entry is a Dictionary with the following keys: `vendor` and `name`. `name` can be set to `*` to add all devices with the specified `vendor`.

Note: This setting is implemented only on Windows.

int **rendering/gl_compatibility/item_buffer_size** = `16384`

Maximum number of canvas items commands that can be drawn in a single viewport update. If more render commands are issued they will be ignored. Decreasing this limit may improve performance on bandwidth limited devices. Increase this limit if you find that not all objects are being drawn in a frame.

bool **rendering/gl_compatibility/nvidia_disable_threaded_optimization** = `true`

If `true`, disables the threaded optimization feature from the NVIDIA drivers, which are known to cause stuttering in most OpenGL applications.

Note: This setting only works on Windows, as threaded optimization is disabled by default on other platforms.

bool **rendering/global_illumination/gi/use_half_resolution** = `false`

If `true`, renders VoxelGI and SDFGI (Environment.sdfgi_enabled) buffers at halved resolution (e.g. 960×540 when the viewport size is 1920×1080). This improves performance significantly when VoxelGI or SDFGI is enabled, at the cost of artifacts that may be visible on polygon edges. The loss in quality becomes less noticeable as the viewport resolution increases. LightmapGI rendering is not affected by this setting.

Note: This property is only read when the project starts. To set half-resolution GI at run-time, call RenderingServer.gi_set_use_half_resolution instead.

int **rendering/global_illumination/sdfgi/frames_to_converge** = `5`

The number of frames to use for converging signed distance field global illumination. Higher values lead to a less noisy result, at the cost of taking a longer time to fully converge. This means the scene's global illumination will be too dark for a longer period of time, especially when the camera moves fast. The actual convergence speed depends on rendered framerate. For example, with the default setting of 30 frames, rendering at 60 FPS will make SDFGI fully converge after 0.5 seconds. See also rendering/global_illumination/sdfgi/frames_to_update_lights and rendering/global_illumination/sdfgi/probe_ray_count.

Note: This property is only read when the project starts. To control SDFGI convergence speed at runtime, call RenderingServer.environment_set_sdfgi_frames_to_converge instead.

---

int **rendering/global_illumination/sdfgi/frames_to_update_lights** = `2` 🔗

The number of frames over which dynamic lights should be updated in signed distance field global illumination. Higher values take more time to update indirect lighting coming from dynamic lights, but result in better performance when many dynamic lights are present. See also rendering/global_illumination/sdfgi/frames_to_converge and rendering/global_illumination/sdfgi/probe_ray_count.

Note: This only affects Light3D nodes whose Light3D.light_bake_mode is Light3D.BAKE_DYNAMIC (which is the default). Consider making non-moving lights use the Light3D.BAKE_STATIC bake mode to improve performance.

Note: This property is only read when the project starts. To control SDFGI light update speed at runtime, call RenderingServer.environment_set_sdfgi_frames_to_update_light instead.

---

int **rendering/global_illumination/sdfgi/probe_ray_count** = `1` 🔗

The number of rays to throw per frame when computing signed distance field global illumination. Higher values lead to a less noisy result, at the cost of performance. See also rendering/global_illumination/sdfgi/frames_to_converge and rendering/global_illumination/sdfgi/frames_to_update_lights.

Note: This property is only read when the project starts. To control SDFGI quality at runtime, call RenderingServer.environment_set_sdfgi_ray_count instead.

---

int **rendering/global_illumination/voxel_gi/quality** = `0` 🔗

The VoxelGI quality to use. High quality leads to more precise lighting and better reflections, but is slower to render. This setting does not affect the baked data and doesn't require baking the VoxelGI again to apply.

Note: This property is only read when the project starts. To control VoxelGI quality at runtime, call RenderingServer.voxel_gi_set_quality instead.

---

int **rendering/lightmapping/bake_performance/max_rays_per_pass** = `32` 🔗

The maximum number of rays that can be thrown per pass when baking lightmaps with LightmapGI. Depending on the scene, adjusting this value may result in higher GPU utilization when baking lightmaps, leading to faster bake times.

---

int **rendering/lightmapping/bake_performance/max_rays_per_probe_pass** = `64` 🔗

The maximum number of rays that can be thrown per pass when baking dynamic object lighting in LightmapProbes with LightmapGI. Depending on the scene, adjusting this value may result in higher GPU utilization when baking lightmaps, leading to faster bake times.

---

int **rendering/lightmapping/bake_performance/region_size** = `512`

The region size to use when baking lightmaps with LightmapGI.

---

int **rendering/lightmapping/bake_quality/high_quality_probe_ray_count** = `512`

The number of rays to use for baking dynamic object lighting in LightmapProbes when LightmapGI.quality is LightmapGI.BAKE_QUALITY_HIGH.

---

int **rendering/lightmapping/bake_quality/high_quality_ray_count** = `512`

The number of rays to use for baking lightmaps with LightmapGI when LightmapGI.quality is LightmapGI.BAKE_QUALITY_HIGH.

---

int **rendering/lightmapping/bake_quality/low_quality_probe_ray_count** = `64`

The number of rays to use for baking dynamic object lighting in LightmapProbes when LightmapGI.quality is LightmapGI.BAKE_QUALITY_LOW.

---

int **rendering/lightmapping/bake_quality/low_quality_ray_count** = `32`

The number of rays to use for baking lightmaps with LightmapGI when LightmapGI.quality is LightmapGI.BAKE_QUALITY_LOW.

---

int **rendering/lightmapping/bake_quality/medium_quality_probe_ray_count** = `256`

The number of rays to use for baking dynamic object lighting in LightmapProbes when LightmapGI.quality is LightmapGI.BAKE_QUALITY_MEDIUM.

---

int **rendering/lightmapping/bake_quality/medium_quality_ray_count** = `128`

The number of rays to use for baking lightmaps with LightmapGI when LightmapGI.quality is LightmapGI.BAKE_QUALITY_MEDIUM.

---

int **rendering/lightmapping/bake_quality/ultra_quality_probe_ray_count** = `2048`

The number of rays to use for baking dynamic object lighting in LightmapProbes when LightmapGI.quality is LightmapGI.BAKE_QUALITY_ULTRA.

---

int **rendering/lightmapping/bake_quality/ultra_quality_ray_count** = `2048`

The number of rays to use for baking lightmaps with LightmapGI when LightmapGI.quality is LightmapGI.BAKE_QUALITY_ULTRA.

---

int **rendering/lightmapping/denoising/denoiser** = `0`

Denoiser tool used for denoising lightmaps.

Using OpenImageDenoise [https://www.openimagedenoise.org/] (OIDN) requires configuring a path to an OIDN executable in the editor settings at EditorSettings.filesystem/tools/oidn/oidn_denoise_path. OIDN can be downloaded from OpenImageDenoise's downloads page [https://www.openimagedenoise.org/downloads.html].

OIDN will use GPU acceleration when available. Unlike JNLM which uses compute shaders for acceleration, OIDN uses vendor-specific acceleration methods. For GPU acceleration to be available, the following libraries must be installed on the system depending on your GPU:

- NVIDIA GPUs: CUDA libraries
- AMD GPUs: HIP libraries
- Intel GPUs: SYCL libraries

If no GPU acceleration is configured on the system, multi-threaded CPU-based denoising will be performed instead. This CPU-based denoising is significantly slower than the JNLM denoiser in most cases.

float **rendering/lightmapping/primitive_meshes/texel_size** = `0.2` 🔗

The texel_size that is used to calculate the Mesh.lightmap_size_hint on PrimitiveMesh resources if PrimitiveMesh.add_uv2 is enabled.

float **rendering/lightmapping/probe_capture/update_speed** = `15` 🔗

The framerate-independent update speed when representing dynamic object lighting from LightmapProbes. Higher values make dynamic object lighting update faster. Higher values can prevent fast-moving objects from having "outdated" indirect lighting displayed on them, at the cost of possible flickering when an object moves from a bright area to a shaded area.

bool **rendering/lights_and_shadows/directional_shadow/16_bits** = `true` 🔗

Use 16 bits for the directional shadow depth map. Enabling this results in shadows having less precision and may result in shadow acne, but can lead to performance improvements on some devices.

int **rendering/lights_and_shadows/directional_shadow/size** = `4096` 🔗

The directional shadow's size in pixels. Higher values will result in sharper shadows, at the cost of performance. The value is rounded up to the nearest power of 2.

int **rendering/lights_and_shadows/directional_shadow/size.mobile** = `2048` 🔗

Lower-end override for rendering/lights_and_shadows/directional_shadow/size on mobile devices, due to performance concerns or driver support.

int **rendering/lights_and_shadows/directional_shadow/soft_shadow_filter_quality** = `2` 🔗

Quality setting for shadows cast by DirectionalLight3Ds. Higher quality settings use more samples when reading from shadow maps and are thus slower. Low quality settings may result in shadows looking grainy.

Note: The Soft Very Low setting will automatically multiply *constant* shadow blur by 0.75x to reduce the amount of noise visible. This automatic blur change only affects the constant blur factor defined in Light3D.shadow_blur, not the variable blur performed by DirectionalLight3Ds' Light3D.light_angular_distance.

Note: The Soft High and Soft Ultra settings will automatically multiply *constant* shadow blur by 1.5× and 2× respectively to make better use of the increased sample count. This increased blur also improves stability of dynamic object shadows.

int **rendering/lights_and_shadows/directional_shadow/soft_shadow_filter_quality.mobile** = `0`

Lower-end override for rendering/lights_and_shadows/directional_shadow/soft_shadow_filter_quality on mobile devices, due to performance concerns or driver support.

bool **rendering/lights_and_shadows/positional_shadow/atlas_16_bits** = `true`

Use 16 bits for the omni/spot shadow depth map. Enabling this results in shadows having less precision and may result in shadow acne, but can lead to performance improvements on some devices.

int **rendering/lights_and_shadows/positional_shadow/atlas_quadrant_0_subdiv** = `2`

Subdivision quadrant size for shadow mapping. See shadow mapping documentation.

int **rendering/lights_and_shadows/positional_shadow/atlas_quadrant_1_subdiv** = `2`

Subdivision quadrant size for shadow mapping. See shadow mapping documentation.

int **rendering/lights_and_shadows/positional_shadow/atlas_quadrant_2_subdiv** = `3`

Subdivision quadrant size for shadow mapping. See shadow mapping documentation.

int **rendering/lights_and_shadows/positional_shadow/atlas_quadrant_3_subdiv** = `4`

Subdivision quadrant size for shadow mapping. See shadow mapping documentation.

int **rendering/lights_and_shadows/positional_shadow/atlas_size** = `4096`

Size for shadow atlas (used for OmniLights and SpotLights). See documentation.

int **rendering/lights_and_shadows/positional_shadow/atlas_size.mobile** = `2048`

Lower-end override for rendering/lights_and_shadows/positional_shadow/atlas_size on mobile devices, due to performance concerns or driver support.

---

int **rendering/lights_and_shadows/positional_shadow/soft_shadow_filter_quality** = `2` 🔗

Quality setting for shadows cast by OmniLight3Ds and SpotLight3Ds. Higher quality settings use more samples when reading from shadow maps and are thus slower. Low quality settings may result in shadows looking grainy.

Note: The Soft Very Low setting will automatically multiply *constant* shadow blur by 0.75x to reduce the amount of noise visible. This automatic blur change only affects the constant blur factor defined in Light3D.shadow_blur, not the variable blur performed by DirectionalLight3Ds' Light3D.light_angular_distance.

Note: The Soft High and Soft Ultra settings will automatically multiply shadow blur by 1.5× and 2× respectively to make better use of the increased sample count. This increased blur also improves stability of dynamic object shadows.

---

int **rendering/lights_and_shadows/positional_shadow/soft_shadow_filter_quality.mobile** = `0` 🔗

Lower-end override for rendering/lights_and_shadows/positional_shadow/soft_shadow_filter_quality on mobile devices, due to performance concerns or driver support.

---

bool **rendering/lights_and_shadows/tighter_shadow_caster_culling** = `true` 🔗

If `true`, items that cannot cast shadows into the view frustum will not be rendered into shadow maps.

This can increase performance.

---

bool **rendering/lights_and_shadows/use_physical_light_units** = `false` 🔗

Enables the use of physically based units for light sources. Physically based units tend to be much larger than the arbitrary units used by Godot, but they can be used to match lighting within Godot to real-world lighting. Due to the large dynamic range of lighting conditions present in nature, Godot bakes exposure into the various lighting quantities before rendering. Most light sources bake exposure automatically at run time based on the active CameraAttributes resource, but LightmapGI and VoxelGI require a CameraAttributes resource to be set at bake time to reduce the dynamic range. At run time, Godot will automatically reconcile the baked exposure with the active exposure to ensure lighting remains consistent.

---

float **rendering/limits/cluster_builder/max_clustered_elements** = `512` 🔗

The maximum number of clustered elements (OmniLight3D + SpotLight3D + Decal + ReflectionProbe) that can be rendered at once in the camera view. If there are more clustered elements present in the camera view, some of them will not be rendered (leading to pop-in during camera movement). Enabling distance fade on lights and decals (Light3D.distance_fade_enabled, Decal.distance_fade_enabled) can help avoid reaching this limit.

Decreasing this value may improve GPU performance on certain setups, even if the maximum number of clustered elements is never reached in the project.

Note: This setting is only effective when using the Forward+ rendering method, not Mobile and Compatibility.

int **rendering/limits/global_shader_variables/buffer_size** = `65536`

The maximum number of uniforms that can be used by the global shader uniform buffer. Each item takes up one slot. In other words, a single uniform float and a uniform vec4 will take the same amount of space in the buffer.

Note: When using the Compatibility backend, most mobile devices (and all web exports) will be limited to a maximum size of 1024 due to hardware constraints.

int **rendering/limits/opengl/max_lights_per_object** = `8`

Max number of omnilights and spotlights renderable per object. At the default value of 8, this means that each surface can be affected by up to 8 omnilights and 8 spotlights. This is further limited by hardware support and rendering/limits/opengl/max_renderable_lights. Setting this low will slightly reduce memory usage, may decrease shader compile times, and may result in faster rendering on low-end, mobile, or web devices.

Note: This setting is only effective when using the Compatibility rendering method, not Forward+ and Mobile.

int **rendering/limits/opengl/max_renderable_elements** = `65536`

Max number of elements renderable in a frame. If more elements than this are visible per frame, they will not be drawn. Keep in mind elements refer to mesh surfaces and not meshes themselves. Setting this low will slightly reduce memory usage and may decrease shader compile times, particularly on web. For most uses, the default value is suitable, but consider lowering as much as possible on web export.

Note: This setting is only effective when using the Compatibility rendering method, not Forward+ and Mobile.

int **rendering/limits/opengl/max_renderable_lights** = `32`

Max number of positional lights renderable in a frame. If more lights than this number are used, they will be ignored. Setting this low will slightly reduce memory usage and may decrease shader compile times, particularly on web. For most uses, the default value is suitable, but consider lowering as much as possible on web export.

Note: This setting is only effective when using the Compatibility rendering method, not Forward+ and Mobile.

int **rendering/limits/spatial_indexer/threaded_cull_minimum_instances** = `1000`

The minimum number of instances that must be present in a scene to enable culling computations on multiple threads. If a scene has fewer instances than this number, culling is done on a single thread.

int **rendering/limits/spatial_indexer/update_iterations_per_frame** = `10`

There is currently no description for this property. Please help us by contributing one!

---

float **rendering/limits/time/time_rollover_secs** = `3600`

There is currently no description for this property. Please help us by contributing one!

---

float **rendering/mesh_lod/lod_change/threshold_pixels** = `1.0`

The automatic LOD bias to use for meshes rendered within the ReflectionProbe. Higher values will use less detailed versions of meshes that have LOD variations generated. If set to `0.0`, automatic LOD is disabled. Increase rendering/mesh_lod/lod_change/threshold_pixels to improve performance at the cost of geometry detail.

Note: rendering/mesh_lod/lod_change/threshold_pixels does not affect GeometryInstance3D visibility ranges (also known as "manual" LOD or hierarchical LOD).

Note: This property is only read when the project starts. To adjust the automatic LOD threshold at runtime, set Viewport.mesh_lod_threshold on the root Viewport.

---

int **rendering/occlusion_culling/bvh_build_quality** = `2`

The Bounding Volume Hierarchy [https://en.wikipedia.org/wiki/Bounding_volume_hierarchy] quality to use when rendering the occlusion culling buffer. Higher values will result in more accurate occlusion culling, at the cost of higher CPU usage. See also rendering/occlusion_culling/occlusion_rays_per_thread.

Note: This property is only read when the project starts. To adjust the BVH build quality at runtime, use RenderingServer.viewport_set_occlusion_culling_build_quality.

---

bool **rendering/occlusion_culling/jitter_projection** = `true`

If `true`, the projection used for rendering the occlusion buffer will be jittered. This can help prevent objects being incorrectly culled when visible through small gaps.

---

int **rendering/occlusion_culling/occlusion_rays_per_thread** = `512`

The number of occlusion rays traced per CPU thread. Higher values will result in more accurate occlusion culling, at the cost of higher CPU usage. The occlusion culling buffer's pixel count is roughly equal to `occlusion_rays_per_thread * number_of_logical_cpu_cores`, so it will depend on the system's CPU. Therefore, CPUs with fewer cores will use a lower resolution to attempt keeping performance costs even across devices. See also rendering/occlusion_culling/bvh_build_quality.

Note: This property is only read when the project starts. To adjust the number of occlusion rays traced per thread at runtime, use RenderingServer.viewport_set_occlusion_rays_per_thread.

---

bool **rendering/occlusion_culling/use_occlusion_culling** = `false`

If `true`, OccluderInstance3D nodes will be usable for occlusion culling in 3D in the root viewport. In custom viewports, Viewport.use_occlusion_culling must be set to `true` instead.

Note: Enabling occlusion culling has a cost on the CPU. Only enable occlusion culling if you actually plan to use it. Large open scenes with few or no objects blocking the view will generally not benefit much from occlusion culling. Large open scenes generally benefit more from mesh LOD and visibility ranges (GeometryInstance3D.visibility_range_begin and GeometryInstance3D.visibility_range_end) compared to occlusion culling.

Note: Due to memory constraints, occlusion culling is not supported by default in Web export templates. It can be enabled by compiling custom Web export templates with `module_raycast_enabled=yes`.

---

int **rendering/reflections/reflection_atlas/reflection_count** = `64` 🔗

Number of cubemaps to store in the reflection atlas. The number of ReflectionProbes in a scene will be limited by this amount. A higher number requires more VRAM.

---

int **rendering/reflections/reflection_atlas/reflection_size** = `256` 🔗

Size of cubemap faces for ReflectionProbes. A higher number requires more VRAM and may make reflection probe updating slower.

---

int **rendering/reflections/reflection_atlas/reflection_size.mobile** = `128` 🔗

Lower-end override for rendering/reflections/reflection_atlas/reflection_size on mobile devices, due to performance concerns or driver support.

---

bool **rendering/reflections/sky_reflections/fast_filter_high_quality** = `false` 🔗

Use a higher quality variant of the fast filtering algorithm. Significantly slower than using default quality, but results in smoother reflections. Should only be used when the scene is especially detailed.

---

int **rendering/reflections/sky_reflections/ggx_samples** = `32` 🔗

Sets the number of samples to take when using importance sampling for Skys and ReflectionProbes. A higher value will result in smoother, higher quality reflections, but increases time to calculate radiance maps. In general, fewer samples are needed for simpler, low dynamic range environments while more samples are needed for HDR environments and environments with a high level of detail.

---

int **rendering/reflections/sky_reflections/ggx_samples.mobile** = `16` 🔗

Lower-end override for rendering/reflections/sky_reflections/ggx_samples on mobile devices, due to performance concerns or driver support.

---

int **rendering/reflections/sky_reflections/roughness_layers** = `8` 🔗

Limits the number of layers to use in radiance maps when using importance sampling. A lower number will be slightly faster and take up less VRAM.

---

bool **rendering/reflections/sky_reflections/texture_array_reflections** = `true`

If `true`, uses texture arrays instead of mipmaps for reflection probes and panorama backgrounds (sky). This reduces jitter noise and upscaling artifacts on reflections, but is significantly slower to compute and uses rendering/reflections/sky_reflections/roughness_layers times more memory.

---

bool **rendering/reflections/sky_reflections/texture_array_reflections.mobile** = `false`

Lower-end override for rendering/reflections/sky_reflections/texture_array_reflections on mobile devices, due to performance concerns or driver support.

String **rendering/renderer/rendering_method** = `"forward_plus"`

Sets the renderer that will be used by the project. Options are:

Forward Plus: High-end renderer designed for Desktop devices. Has a higher base overhead, but scales well with complex scenes. Not suitable for older devices or mobile.

Mobile: Modern renderer designed for mobile devices. Has a lower base overhead than Forward Plus, but does not scale as well to large scenes with many elements.

GL Compatibility: Low-end renderer designed for older devices. Based on the limitations of the OpenGL 3.3/ OpenGL ES 3.0 / WebGL 2 APIs.

String **rendering/renderer/rendering_method.mobile** = `"mobile"`

Override for rendering/renderer/rendering_method on mobile devices.

String **rendering/renderer/rendering_method.web** = `"gl_compatibility"`

Override for rendering/renderer/rendering_method on web.

---

int **rendering/rendering_device/d3d12/agility_sdk_version** = `613`

Version code of the Direct3D 12 Agility SDK [https://devblogs.microsoft.com/directx/directx12agility/] to use (`D3D12SDKVersion`). This must match the *minor* version that is installed next to the editor binary and in the export templates directory for the current editor version. For example, if you have `1.613.3` installed, you need to input `613` here.

---

int **rendering/rendering_device/d3d12/max_misc_descriptors_per_frame** = `512`

The number of entries in the miscellaneous descriptors heap the Direct3D 12 rendering driver uses each frame, used for various operations like clearing a texture.

Depending on the complexity of scenes, this value may be lowered or may need to be raised.

---

int **rendering/rendering_device/d3d12/max_resource_descriptors_per_frame** = `16384`

The number of entries in the resource descriptors heap the Direct3D 12 rendering driver uses each frame, used for most rendering operations.

Depending on the complexity of scenes, this value may be lowered or may need to be raised.

---

int **rendering/rendering_device/d3d12/max_sampler_descriptors_per_frame** = `1024`

The number of entries in the sampler descriptors heap the Direct3D 12 rendering driver uses each frame, used for most rendering operations.

Depending on the complexity of scenes, this value may be lowered or may need to be raised.

String **rendering/rendering_device/driver**

Sets the driver to be used by the renderer when using a RenderingDevice-based renderer like the clustered renderer or the mobile renderer. This property can not be edited directly, instead, set the driver using the platform-specific overrides.

String **rendering/rendering_device/driver.android**

Android override for rendering/rendering_device/driver.

String **rendering/rendering_device/driver.ios**

iOS override for rendering/rendering_device/driver.

String **rendering/rendering_device/driver.linuxbsd**

LinuxBSD override for rendering/rendering_device/driver.

String **rendering/rendering_device/driver.macos**

macOS override for rendering/rendering_device/driver.

String **rendering/rendering_device/driver.windows**

Windows override for rendering/rendering_device/driver.

---

bool **rendering/rendering_device/fallback_to_d3d12** = `true`

If `true`, the forward renderer will fall back to Direct3D 12 if Vulkan is not supported.

Note: This setting is implemented only on Windows.

---

bool **rendering/rendering_device/fallback_to_vulkan** = `true`

If `true`, the forward renderer will fall back to Vulkan if Direct3D 12 is not supported.

Note: This setting is implemented only on Windows.

---

bool **rendering/rendering_device/pipeline_cache/enable** = `true`

Enable the pipeline cache that is saved to disk if the graphics API supports it.

Note: This property is unable to control the pipeline caching the GPU driver itself does. Only turn this off along with deleting the contents of the driver's cache if you wish to simulate the experience a user will get when starting the game for the first time.

---

float **rendering/rendering_device/pipeline_cache/save_chunk_size_mb** = `3.0`

Determines at which interval pipeline cache is saved to disk. The lower the value, the more often it is saved.

---

int **rendering/rendering_device/staging_buffer/block_size_kb** = `256`

There is currently no description for this property. Please help us by contributing one!

---

int **rendering/rendering_device/staging_buffer/max_size_mb** = `128`

There is currently no description for this property. Please help us by contributing one!

---

int **rendering/rendering_device/staging_buffer/texture_upload_region_size_px** = `64`

There is currently no description for this property. Please help us by contributing one!

---

int **rendering/rendering_device/vsync/frame_queue_size** = `2`

The number of frames to track on the CPU side before stalling to wait for the GPU.

Try the V-Sync Simulator [https://darksylinc.github.io/vsync_simulator/], an interactive interface that simulates presentation to better understand how it is affected by different variables under various conditions.

Note: This property is only read when the project starts. There is currently no way to change this value at run-time.

---

int **rendering/rendering_device/vsync/swapchain_image_count** = `3`

The number of images the swapchain will consist of (back buffers + front buffer).

`2` corresponds to double-buffering and `3` to triple-buffering.

Double-buffering may give you the lowest lag/latency but if V-Sync is on and the system can't render at 60 fps, the framerate will go down in multiples of it (e.g. 30 fps, 15, 7.5, etc.). Triple buffering gives you higher framerate (specially if the system can't reach a constant 60 fps) at the cost of up to 1 frame of latency, with DisplayServer.VSYNC_ENABLED (FIFO).

Use double-buffering with DisplayServer.VSYNC_ENABLED. Triple-buffering is a must if you plan on using DisplayServer.VSYNC_MAILBOX mode.

Try the V-Sync Simulator [https://darksylinc.github.io/vsync_simulator/], an interactive interface that simulates presentation to better understand how it is affected by different variables under various conditions.

Note: This property is only read when the project starts. There is currently no way to change this value at run-time.

Note: Some platforms may restrict the actual value.

---

int **rendering/rendering_device/vulkan/max_descriptors_per_pool** = `64`

There is currently no description for this property. Please help us by contributing one!

---

float **rendering/scaling_3d/fsr_sharpness** = `0.2`

Determines how sharp the upscaled image will be when using the FSR upscaling mode. Sharpness halves with every whole number. Values go from 0.0 (sharpest) to 2.0. Values above 2.0 won't make a visible difference.

---

int **rendering/scaling_3d/mode** = `0`

Sets the scaling 3D mode. Bilinear scaling renders at different resolution to either undersample or supersample the viewport. FidelityFX Super Resolution 1.0, abbreviated to FSR, is an upscaling technology that produces high quality images at fast framerates by using a spatially-aware upscaling algorithm. FSR is slightly more expensive than bilinear, but it produces significantly higher image quality. On particularly low-end GPUs, the added cost of FSR may not be worth it (compared to using bilinear scaling with a slightly higher resolution scale to match performance).

Note: FSR is only effective when using the Forward+ rendering method, not Mobile or Compatibility. If using an incompatible rendering method, FSR will fall back to bilinear scaling.

---

float **rendering/scaling_3d/scale** = `1.0`

Scales the 3D render buffer based on the viewport size uses an image filter specified in rendering/scaling_3d/mode to scale the output image to the full viewport size. Values lower than `1.0` can be used to speed up 3D rendering at the cost of quality (undersampling). Values greater than `1.0` are only valid for bilinear mode and can be used to improve 3D rendering quality at a high performance cost (supersampling). See also rendering/anti_aliasing/quality/msaa_3d for multi-sample antialiasing, which is significantly cheaper but only smooths the edges of polygons.

---

bool **rendering/shader_compiler/shader_cache/compress** = `true`

There is currently no description for this property. Please help us by contributing one!

---

bool **rendering/shader_compiler/shader_cache/enabled** = `true`

Enable the shader cache, which stores compiled shaders to disk to prevent stuttering from shader compilation the next time the shader is needed.

---

bool **rendering/shader_compiler/shader_cache/strip_debug** = `false`

There is currently no description for this property. Please help us by contributing one!

---

bool **rendering/shader_compiler/shader_cache/strip_debug.release** = `true`

There is currently no description for this property. Please help us by contributing one!

---

bool **rendering/shader_compiler/shader_cache/use_zstd_compression** = `true`

There is currently no description for this property. Please help us by contributing one!

---

bool **rendering/shading/overrides/force_lambert_over_burley** = `false`

If `true`, uses faster but lower-quality Lambert material lighting model instead of Burley.

---

bool **rendering/shading/overrides/force_lambert_over_burley.mobile** = `true`

Lower-end override for rendering/shading/overrides/force_lambert_over_burley on mobile devices, due to performance concerns or driver support.

---

bool **rendering/shading/overrides/force_vertex_shading** = `false`

If `true`, forces vertex shading for all rendering. This can increase performance a lot, but also reduces quality immensely. Can be used to optimize performance on low-end mobile devices.

Note: This setting currently has no effect, as vertex shading is not implemented yet.

---

bool **rendering/shading/overrides/force_vertex_shading.mobile** = `true`

Lower-end override for rendering/shading/overrides/force_vertex_shading on mobile devices, due to performance concerns or driver support.

Note: This setting currently has no effect, as vertex shading is not implemented yet.

---

int **rendering/textures/canvas_textures/default_texture_filter** = `1`

The default texture filtering mode to use on CanvasItems.

Note: For pixel art aesthetics, see also rendering/2d/snap/snap_2d_vertices_to_pixel and rendering/2d/snap/snap_2d_transforms_to_pixel.

---

int **rendering/textures/canvas_textures/default_texture_repeat** = `0`

The default texture repeating mode to use on CanvasItems.

---

int **rendering/textures/decals/filter** = `3`

The filtering quality to use for Decal nodes. When using one of the anisotropic filtering modes, the anisotropic filtering level is controlled by rendering/textures/default_filters/anisotropic_filtering_level.

---

int **rendering/textures/default_filters/anisotropic_filtering_level** = `2`

Sets the maximum number of samples to take when using anisotropic filtering on textures (as a power of two). A higher sample count will result in sharper textures at oblique angles, but is more expensive to compute. A value of `0` forcibly disables anisotropic filtering, even on materials where it is enabled.

The anisotropic filtering level also affects decals and light projectors if they are configured to use anisotropic filtering. See rendering/textures/decals/filter and rendering/textures/light_projectors/filter.

Note: For performance reasons, anisotropic filtering *is not enabled by default* on 2D and 3D materials. For this setting to have an effect in 3D, set BaseMaterial3D.texture_filter to BaseMaterial3D.TEXTURE_FILTER_LINEAR_WITH_MIPMAPS_ANISOTROPIC or BaseMaterial3D.TEXTURE_FILTER_NEAREST_WITH_MIPMAPS_ANISOTROPIC on materials. For this setting to have an effect in 2D, set CanvasItem.texture_filter to CanvasItem.TEXTURE_FILTER_LINEAR_WITH_MIPMAPS_ANISOTROPIC or CanvasItem.TEXTURE_FILTER_NEAREST_WITH_MIPMAPS_ANISOTROPIC on the CanvasItem node displaying the texture (or in CanvasTexture). However, anisotropic filtering is rarely useful in 2D, so only enable it for textures in 2D if it makes a meaningful visual difference.

Note: This property is only read when the project starts. There is currently no way to change this setting at run-time.

---

float **rendering/textures/default_filters/texture_mipmap_bias** = `0.0`

Affects the final texture sharpness by reading from a lower or higher mipmap (also called "texture LOD bias"). Negative values make mipmapped textures sharper but grainier when viewed at a distance, while positive values make mipmapped textures blurrier (even when up close).

Enabling temporal antialiasing (rendering/anti_aliasing/quality/use_taa) will automatically apply a `-0.5` offset to this value, while enabling FXAA (rendering/anti_aliasing/quality/screen_space_aa) will automatically apply a `-0.25` offset to this value. If both TAA and FXAA are enabled at the same time, an offset of `-0.75` is applied to this value.

Note: If rendering/scaling_3d/scale is lower than `1.0` (exclusive), rendering/textures/default_filters/texture_mipmap_bias is used to adjust the automatic mipmap bias which is calculated internally based on the scale factor. The formula for this is `log2(scaling_3d_scale) + mipmap_bias`.

---

bool **rendering/textures/default_filters/use_nearest_mipmap_filter** = `false`

If `true`, uses nearest-neighbor mipmap filtering when using mipmaps (also called "bilinear filtering"), which will result in visible seams appearing between mipmap stages. This may increase performance in mobile as less memory bandwidth is used. If `false`, linear mipmap filtering (also called "trilinear filtering") is used.

Note: This property is only read when the project starts. There is currently no way to change this setting at run-time.

---

int **rendering/textures/light_projectors/filter** = `3` 🔗

The filtering quality to use for OmniLight3D and SpotLight3D projectors. When using one of the anisotropic filtering modes, the anisotropic filtering level is controlled by rendering/textures/default_filters/anisotropic_filtering_level.

---

bool **rendering/textures/lossless_compression/force_png** = `false` 🔗

If `true`, the texture importer will import lossless textures using the PNG format. Otherwise, it will default to using WebP.

---

bool **rendering/textures/vram_compression/import_etc2_astc** = `false` 🔗

If `true`, the texture importer will import VRAM-compressed textures using the Ericsson Texture Compression 2 algorithm for lower quality textures and normal maps and Adaptable Scalable Texture Compression algorithm for high quality textures (in 4×4 block size).

Note: This setting is an override. The texture importer will always import the format the host platform needs, even if this is set to `false`.

Note: Changing this setting does *not* impact textures that were already imported before. To make this setting apply to textures that were already imported, exit the editor, remove the `.godot/imported/` folder located inside the project folder then restart the editor (see application/config/use_hidden_project_data_directory).

---

bool **rendering/textures/vram_compression/import_s3tc_bptc** = `false` 🔗

If `true`, the texture importer will import VRAM-compressed textures using the S3 Texture Compression algorithm (DXT1-5) for lower quality textures and the BPTC algorithm (BC6H and BC7) for high quality textures. This algorithm is only supported on PC desktop platforms and consoles.

Note: This setting is an override. The texture importer will always import the format the host platform needs, even if this is set to `false`.

Note: Changing this setting does *not* impact textures that were already imported before. To make this setting apply to textures that were already imported, exit the editor, remove the `.godot/imported/` folder located inside the project folder then restart the editor (see application/config/use_hidden_project_data_directory).

---

int **rendering/textures/webp_compression/compression_method** = `2` 🔗

The default compression method for WebP. Affects both lossy and lossless WebP. A higher value results in smaller files at the cost of compression speed. Decompression speed is mostly unaffected by the compression method. Supported values are 0 to 6. Note that compression methods above 4 are very slow and offer very little savings.

float **rendering/textures/webp_compression/lossless_compression_factor** = `25`

The default compression factor for lossless WebP. Decompression speed is mostly unaffected by the compression factor. Supported values are 0 to 100.

bool **rendering/viewport/hdr_2d** = `false`

If `true`, enables Viewport.use_hdr_2d on the root viewport. This allows 2D rendering to take advantage of effects requiring high dynamic range (e.g. 2D glow).

bool **rendering/viewport/transparent_background** = `false`

If `true`, enables Viewport.transparent_bg on the root viewport. This allows per-pixel transparency to be effective after also enabling display/window/size/transparent and display/window/per_pixel_transparency/allowed.

int **rendering/vrs/mode** = `0`

Set the default Variable Rate Shading (VRS) mode for the main viewport. See Viewport.vrs_mode to change this at runtime, and VRSMode for possible values.

String **rendering/vrs/texture** = `""`

If rendering/vrs/mode is set to **Texture**, this is the path to default texture loaded as the VRS image.

The texture *must* use a lossless compression format so that colors can be matched precisely. The following VRS densities are mapped to various colors, with brighter colors representing a lower level of shading precision:

```
- 1×1 = rgb(0, 0, 0)     - #000000
- 1×2 = rgb(0, 85, 0)    - #005500
- 2×1 = rgb(85, 0, 0)    - #550000
- 2×2 = rgb(85, 85, 0)   - #555500
- 2×4 = rgb(85, 170, 0)  - #55aa00
- 4×2 = rgb(170, 85, 0)  - #aa5500
- 4×4 = rgb(170, 170, 0) - #aaaa00
- 4×8 = rgb(170, 255, 0) - #aaff00 - Not supported on most hardware
- 8×4 = rgb(255, 170, 0) - #ffaa00 - Not supported on most hardware
- 8×8 = rgb(255, 255, 0) - #ffff00 - Not supported on most hardware
```

float **threading/worker_pool/low_priority_thread_ratio** = `0.3`

The ratio of WorkerThreadPool's threads that will be reserved for low-priority tasks. For example, if 10 threads are available and this value is set to `0.3`, 3 of the worker threads will be reserved for low-priority tasks. The actual value won't exceed the number of CPU cores minus one, and if possible, at least one worker thread will be dedicated to low-priority tasks.

int **threading/worker_pool/max_threads** = `-1`

Maximum number of threads to be used by WorkerThreadPool. Value of `-1` means no limit.

String **xr/openxr/default_action_map** = `"res://openxr_action_map.tres"` 🔗

Action map configuration to load by default.

---

bool **xr/openxr/enabled** = `false` 🔗

If `true`, Godot will setup and initialize OpenXR on startup.

---

int **xr/openxr/environment_blend_mode** = `"0"` 🔗

Specify how OpenXR should blend in the environment. This is specific to certain AR and passthrough devices where camera images are blended in by the XR compositor.

---

bool **xr/openxr/extensions/eye_gaze_interaction** = `false` 🔗

Specify whether to enable eye tracking for this project. Depending on the platform, additional export configuration may be needed.

---

bool **xr/openxr/extensions/hand_interaction_profile** = `false` 🔗

If true the hand interaction profile extension will be activated if supported by the platform.

---

bool **xr/openxr/extensions/hand_tracking** = `true` 🔗

If true we enable the hand tracking extension if available.

---

int **xr/openxr/form_factor** = `"0"` 🔗

Specify whether OpenXR should be configured for an HMD or a hand held device.

---

bool **xr/openxr/foveation_dynamic** = `false` 🔗

If true and foveation is supported, will automatically adjust foveation level based on framerate up to the level set on xr/openxr/foveation_level.

Note: Only works on the Compatibility rendering method.

---

int **xr/openxr/foveation_level** = `"0"` 🔗

Applied foveation level if supported: 0 = off, 1 = low, 2 = medium, 3 = high.

Note: Only works on the Compatibility rendering method. On platforms other than Android, if rendering/anti_aliasing/quality/msaa_3d is enabled, this feature will be disabled.

---

int **xr/openxr/reference_space** = `"1"` 🔗

Specify the default reference space.

bool **xr/openxr/startup_alert** = `true`

If `true`, Godot will display an alert modal when OpenXR initialization fails on startup.

bool **xr/openxr/submit_depth_buffer** = `false`

If `true`, OpenXR will manage the depth buffer and use the depth buffer for advanced reprojection provided this is supported by the XR runtime. Note that some rendering features in Godot can't be used with this feature.

int **xr/openxr/view_configuration** = `"1"`

Specify the view configuration with which to configure OpenXR setting up either Mono or Stereo rendering.

bool **xr/shaders/enabled** = `false`

If `true`, Godot will compile shaders required for XR.

Method Descriptions

void **add_property_info**(hint: Dictionary)

Adds a custom property info to a property. The dictionary must contain:

- `"name"`: String (the property's name)
- `"type"`: int (see Variant.Type)
- optionally `"hint"`: int (see PropertyHint) and `"hint_string"`: String

Example:

GDScript

```
ProjectSettings.set("category/property_name", 0)

var property_info = {
    "name": "category/property_name",
    "type": TYPE_INT,
    "hint": PROPERTY_HINT_ENUM,
    "hint_string": "one,two,three"
}

ProjectSettings.add_property_info(property_info)
```

C#

```
ProjectSettings.Singleton.Set("category/property_name", 0);

var propertyInfo = new Godot.Collections.Dictionary
{
    {"name", "category/propertyName"},
    {"type", (int)Variant.Type.Int},
    {"hint", (int)PropertyHint.Enum},
    {"hint_string", "one,two,three"},
};

ProjectSettings.AddPropertyInfo(propertyInfo);
```

---

void **clear**(name: String)

Clears the whole configuration (not recommended, may break things).

Array[Dictionary] **get_global_class_list**()

Returns an Array of registered global classes. Each global class is represented as a Dictionary that contains the following entries:

- `base` is a name of the base class;
- `class` is a name of the registered global class;
- `icon` is a path to a custom icon of the global class, if it has any;
- `language` is a name of a programming language in which the global class is written;
- `path` is a path to a file containing the global class.

Note: Both the script and the icon paths are local to the project filesystem, i.e. they start with `res://`.

---

int **get_order**(name: String) const

Returns the order of a configuration value (influences when saved to the config file).

Variant **get_setting**(name: String, default_value: Variant = null) const

Returns the value of the setting identified by `name`. If the setting doesn't exist and `default_value` is specified, the value of `default_value` is returned. Otherwise, `null` is returned.

Example:

GDScript

```
print(ProjectSettings.get_setting("application/config/name"))
print(ProjectSettings.get_setting("application/config/custom_description", "No description specified."))
```

C#

```
GD.Print(ProjectSettings.GetSetting("application/config/name"));
GD.Print(ProjectSettings.GetSetting("application/config/custom_description", "No description specified."));
```

Note: This method doesn't take potential feature overrides into account automatically. Use get_setting_with_override to handle seamlessly.

Variant **get_setting_with_override**(name: StringName) const

Similar to get_setting, but applies feature tag overrides if any exists and is valid.

Example:

If the following setting override exists "application/config/name.windows", and the following code is executed:

GDScript

```
print(ProjectSettings.get_setting_with_override("application/config/name"))
```

C#

```
GD.Print(ProjectSettings.GetSettingWithOverride("application/config/name"));
```

Then the overridden setting will be returned instead if the project is running on the *Windows* operating system.

String **globalize_path**(path: String) const

Returns the absolute, native OS path corresponding to the localized `path` (starting with `res://` or `user://`). The returned path will vary depending on the operating system and user preferences. See File paths in Godot projects to see what those paths convert to. See also localize_path.

Note: globalize_path with `res://` will not work in an exported project. Instead, prepend the executable's base directory to the path when running from an exported project:

```
var path = ""
if OS.has_feature("editor"):
    # Running from an editor binary.
    # `path` will contain the absolute path to `hello.txt` located in the project root.
    path = ProjectSettings.globalize_path("res://hello.txt")
else:
    # Running from an exported project.
    # `path` will contain the absolute path to `hello.txt` next to the executable.
    # This is *not* identical to using `ProjectSettings.globalize_path()` with a `res://` path,
    # but is close enough in spirit.
    path = OS.get_executable_path().get_base_dir().path_join("hello.txt")
```

---

bool **has_setting**(name: String) const

Returns `true` if a configuration value is present.

---

bool **load_resource_pack**(pack: String, replace_files: bool = true, offset: int = 0)

Loads the contents of the .pck or .zip file specified by `pack` into the resource filesystem (`res://`). Returns `true` on success.

Note: If a file from `pack` shares the same path as a file already in the resource filesystem, any attempts to load that file will use the file from `pack` unless `replace_files` is set to `false`.

Note: The optional `offset` parameter can be used to specify the offset in bytes to the start of the resource pack. This is only supported for .pck files.

String **localize_path**(path: String) const

Returns the localized path (starting with `res://`) corresponding to the absolute, native OS `path`. See also globalize_path.

Error **save**()

Saves the configuration to the `project.godot` file.

Note: This method is intended to be used by editor plugins, as modified **ProjectSettings** can't be loaded back in the running app. If you want to change project settings in exported projects, use save_custom to save `override.cfg` file.

Error **save_custom**(file: String)

Saves the configuration to a custom file. The file extension must be `.godot` (to save in text-based ConfigFile format) or `.binary` (to save in binary format). You can also save `override.cfg` file, which is also text, but can be used in exported projects unlike other formats.

---

void **set_as_basic**(name: String, basic: bool)

Defines if the specified setting is considered basic or advanced. Basic settings will always be shown in the project settings. Advanced settings will only be shown if the user enables the "Advanced Settings" option.

---

void **set_as_internal**(name: String, internal: bool)

Defines if the specified setting is considered internal. An internal setting won't show up in the Project Settings dialog. This is mostly useful for addons that need to store their own internal settings without exposing them directly to the user.

---

void **set_initial_value**(name: String, value: Variant)

Sets the specified setting's initial value. This is the value the setting reverts to.

---

void **set_order**(name: String, position: int)

Sets the order of a configuration value (influences when saved to the config file).

---

void **set_restart_if_changed**(name: String, restart: bool)

Sets whether a setting requires restarting the editor to properly take effect.

Note: This is just a hint to display to the user that the editor must be restarted for changes to take effect. Enabling set_restart_if_changed does *not* delay the setting being set when changed.

---

void **set_setting**(name: String, value: Variant)

Sets the value of a setting.

Example:

GDScript

```
ProjectSettings.set_setting("application/config/name", "Example")
```

C#

```
ProjectSettings.SetSetting("application/config/name", "Example");
```

This can also be used to erase custom project settings. To do this change the setting value to `null`.

PropertyTweener

Inherits: Tweener **<** RefCounted **<** Object

Interpolates an Object's property over time.

Description

PropertyTweener is used to interpolate a property in an object. See Tween.tween_property for more usage information.

Note: Tween.tween_property is the only correct way to create **PropertyTweener**. Any **PropertyTweener** created manually will not function correctly.

Methods

PropertyTweener	as_relative()
PropertyTweener	from(value: Variant)
PropertyTweener	from_current()
PropertyTweener	set_custom_interpolator(interpolator_method: Callable)
PropertyTweener	set_delay(delay: float)

PropertyTweener set_ease(ease: EaseType)

PropertyTweener set_trans(trans: TransitionType)

Method Descriptions

PropertyTweener **as_relative**() 🔗

When called, the final value will be used as a relative value instead.

Example:

```
var tween = get_tree().create_tween()
tween.tween_property(self, "position", Vector2.RIGHT * 100,
1).as_relative() #the node will move by 100 pixels to the right
```

PropertyTweener **from**(value: Variant) 🔗

Sets a custom initial value to the **PropertyTweener**.

Example:

```
var tween = get_tree().create_tween()
tween.tween_property(self, "position", Vector2(200, 100),
1).from(Vector2(100, 100)) #this will move the node from
position (100, 100) to (200, 100)
```

PropertyTweener **from_current**() 🔗

Makes the **PropertyTweener** use the current property value (i.e. at the time of creating this **PropertyTweener**) as a starting point.

This is equivalent of using from with the current value. These two calls will do the same:

```
tween.tween_property(self, "position", Vector2(200, 100),
1).from(position)
tween.tween_property(self, "position", Vector2(200, 100),
1).from_current()
```

PropertyTweener **set_custom_interpolator**(interpolator_method: Callable) 🔗

Allows interpolating the value with a custom easing function. The provided `interpolator_method` will be called with a value ranging from `0.0` to `1.0` and is expected to return a value within the same range (values outside the range can be used for overshoot). The return value of the method is then used for interpolation between initial and final value. Note that the parameter passed to the method is still subject to the tweener's own easing.

Example:

```
@export var curve: Curve

func _ready():
    var tween = create_tween()
    # Interpolate the value using a custom curve.
    tween.tween_property(self, "position:x", 300,
1).as_relative().set_custom_interpolator(tween_curve)

func tween_curve(v):
    return curve.sample_baked(v)
```

PropertyTweener **set_delay**(delay: float) 🔗

Sets the time in seconds after which the **PropertyTweener** will start interpolating. By default there's no delay.

PropertyTweener **set_ease**(ease: EaseType)

Sets the type of used easing from EaseType. If not set, the default easing is used from the Tween that contains this Tweener.

PropertyTweener **set_trans**(trans: TransitionType)

Sets the type of used transition from TransitionType. If not set, the default transition is used from the Tween that contains this Tweener.

RandomNumberGenerator

Inherits: RefCounted **<** Object

Provides methods for generating pseudo-random numbers.

Description

RandomNumberGenerator is a class for generating pseudo-random numbers. It currently uses PCG32 [https://www.pcg-random.org/].

Note: The underlying algorithm is an implementation detail and should not be depended upon.

To generate a random float number (within a given range) based on a time-dependent seed:

```
var rng = RandomNumberGenerator.new()
func _ready():
    var my_random_number = rng.randf_range(-10.0, 10.0)
```

Tutorials

- Random number generation

Properties

int	seed	0
int	state	0

Methods

int	rand_weighted(weights: PackedFloat32Array)
float	randf()
float	randf_range(from: float, to: float)
float	randfn(mean: float = 0.0, deviation: float = 1.0)
int	randi()
int	randi_range(from: int, to: int)
void	randomize()

Property Descriptions

int **seed** = 0 🔗

- void **set_seed**(value: int)
- int **get_seed**()

Initializes the random number generator state based on the given seed value. A given seed will give a reproducible sequence of pseudo-random numbers.

Note: The RNG does not have an avalanche effect, and can output similar random streams given similar seeds. Consider using a hash function to improve your seed quality if they're sourced externally.

Note: Setting this property produces a side effect of changing the internal state, so make sure to initialize the seed *before* modifying the state:

Note: The default value of this property is pseudo-random, and changes when calling randomize. The `0` value documented here is a placeholder, and not the actual default seed.

```
var rng = RandomNumberGenerator.new()
rng.seed = hash("Godot")
rng.state = 100 # Restore to some previously saved state.
```

---

int **state** = `0` 🔗

- void **set_state**(value: int)
- int **get_state**()

The current state of the random number generator. Save and restore this property to restore the generator to a previous state:

```
var rng = RandomNumberGenerator.new()
print(rng.randf())
var saved_state = rng.state # Store current state.
print(rng.randf()) # Advance internal state.
rng.state = saved_state # Restore the state.
print(rng.randf()) # Prints the same value as in previous.
```

Note: Do not set state to arbitrary values, since the random number generator requires the state to have certain qualities to behave properly. It should only be set to values that came from the state property itself. To initialize the random number generator with arbitrary input, use seed instead.

Note: The default value of this property is pseudo-random, and changes when calling randomize. The `0` value documented here is a placeholder, and not the actual default seed.

Method Descriptions

int **rand_weighted**(weights: PackedFloat32Array)

Returns a random index with non-uniform weights. Prints an error and returns `-1` if the array is empty.

GDScript

```
var rng = RandomNumberGenerator.new()

var my_array = ["one", "two", "three", "four"]
var weights = PackedFloat32Array([0.5, 1, 1, 2])

# Prints one of the four elements in `my_array`.
# It is more likely to print "four", and less likely to print "one".
print(my_array[rng.rand_weighted(weights)])
```

float **randf**()

Returns a pseudo-random float between `0.0` and `1.0` (inclusive).

float **randf_range**(from: float, to: float)

Returns a pseudo-random float between `from` and `to` (inclusive).

float **randfn**(mean: float = 0.0, deviation: float = 1.0)

Returns a normally-distributed [https://en.wikipedia.org/wiki/Normal_distribution], pseudo-random floating-point number from the specified `mean` and a standard `deviation`. This is also known as a Gaussian distribution.

Note: This method uses the Box-Muller transform [https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform] algorithm.

---

int **randi**() 🔗

Returns a pseudo-random 32-bit unsigned integer between `0` and `4294967295` (inclusive).

---

int **randi_range**(from: int, to: int) 🔗

Returns a pseudo-random 32-bit signed integer between `from` and `to` (inclusive).

---

void **randomize**() 🔗

Sets up a time-based seed for this **RandomNumberGenerator** instance. Unlike the @GlobalScope random number generation functions, different **RandomNumberGenerator** instances can use different seeds.

RDAttachmentFormat

Inherits: RefCounted **<** Object

Attachment format (used by RenderingDevice).

Description

This object is used by RenderingDevice.

Properties

DataFormat	format	`36`
TextureSamples	samples	`0`
int	usage_flags	`0`

Property Descriptions

DataFormat **format** = `36` 🔗

- void **set_format**(value: DataFormat)
- DataFormat **get_format**()

The attachment's data format.

TextureSamples **samples** = 0 🔗

- void **set_samples**(value: TextureSamples)
- TextureSamples **get_samples**()

The number of samples used when sampling the attachment.

---

int **usage_flags** = 0 🔗

- void **set_usage_flags**(value: int)
- int **get_usage_flags**()

The attachment's usage flags, which determine what can be done with it.

RDFramebufferPass

Inherits: RefCounted **<** Object

Framebuffer pass attachment description (used by RenderingDevice).

Description

This class contains the list of attachment descriptions for a framebuffer pass. Each points with an index to a previously supplied list of texture attachments.

Multipass framebuffers can optimize some configurations in mobile. On desktop, they provide little to no advantage.

This object is used by RenderingDevice.

Properties

PackedInt32Array	color_attachments	`PackedInt32Array()`
int	depth_attachment	`-1`
PackedInt32Array	input_attachments	`PackedInt32Array()`
PackedInt32Array	preserve_attachments	`PackedInt32Array()`

PackedInt32Array	resolve_attachments	`PackedInt32Array()`

Constants

ATTACHMENT_UNUSED = `-1`

Attachment is unused.

Property Descriptions

PackedInt32Array **color_attachments** = `PackedInt32Array()`

- void **set_color_attachments**(value: PackedInt32Array)
- PackedInt32Array **get_color_attachments**()

Color attachments in order starting from 0. If this attachment is not used by the shader, pass ATTACHMENT_UNUSED to skip.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedInt32Array for more details.

int **depth_attachment** = `-1`

- void **set_depth_attachment**(value: int)
- int **get_depth_attachment**()

Depth attachment. ATTACHMENT_UNUSED should be used if no depth buffer is required for this pass.

PackedInt32Array **input_attachments** = `PackedInt32Array()` 🔗

- void **set_input_attachments**(value: PackedInt32Array)
- PackedInt32Array **get_input_attachments**()

Used for multipass framebuffers (more than one render pass). Converts an attachment to an input. Make sure to also supply it properly in the RDUniform for the uniform set.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedInt32Array for more details.

PackedInt32Array **preserve_attachments** = `PackedInt32Array()` 🔗

- void **set_preserve_attachments**(value: PackedInt32Array)
- PackedInt32Array **get_preserve_attachments**()

Attachments to preserve in this pass (otherwise they are erased).

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedInt32Array for more details.

PackedInt32Array **resolve_attachments** = `PackedInt32Array()` 🔗

- void **set_resolve_attachments**(value: PackedInt32Array)
- PackedInt32Array **get_resolve_attachments**()

If the color attachments are multisampled, non-multisampled resolve attachments can be provided.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedInt32Array for more details.

RDPipelineColorBlendState

Inherits: RefCounted < Object

Pipeline color blend state (used by RenderingDevice).

Description

This object is used by RenderingDevice.

Properties

Array[RDPipelineColorBlendStateAttachment]	attachments	`[]`
Color	blend_constant	`Color(0, 0, 0, 1)`
bool	enable_logic_op	`false`
LogicOperation	logic_op	`0`

Property Descriptions

Array[RDPipelineColorBlendStateAttachment] **attachments** = `[]` 🔗

- void **set_attachments**(value: Array[RDPipelineColorBlendStateAttachment])
- Array[RDPipelineColorBlendStateAttachment] **get_attachments**()

The attachments that are blended together.

Color **blend_constant** = `Color(0, 0, 0, 1)`

- void **set_blend_constant**(value: Color)
- Color **get_blend_constant**()

The constant color to blend with. See also RenderingDevice.draw_list_set_blend_constants.

---

bool **enable_logic_op** = `false`

- void **set_enable_logic_op**(value: bool)
- bool **get_enable_logic_op**()

If `true`, performs the logic operation defined in logic_op.

LogicOperation **logic_op** = `0`

- void **set_logic_op**(value: LogicOperation)
- LogicOperation **get_logic_op**()

The logic operation to perform for blending. Only effective if enable_logic_op is `true`.

RDPipelineColorBlendStateAttachment

Inherits: RefCounted **<** Object

Pipeline color blend state attachment (used by RenderingDevice).

Description

Controls how blending between source and destination fragments is performed when using RenderingDevice.

For reference, this is how common user-facing blend modes are implemented in Godot's 2D renderer:

Mix:

```
var attachment = RDPipelineColorBlendStateAttachment.new()
attachment.enable_blend = true
attachment.color_blend_op = RenderingDevice.BLEND_OP_ADD
attachment.src_color_blend_factor =
RenderingDevice.BLEND_FACTOR_SRC_ALPHA
attachment.dst_color_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE_MINUS_SRC_ALPHA
attachment.alpha_blend_op = RenderingDevice.BLEND_OP_ADD
attachment.src_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE
attachment.dst_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE_MINUS_SRC_ALPHA
```

Add:

```
var attachment = RDPipelineColorBlendStateAttachment.new()
attachment.enable_blend = true
attachment.alpha_blend_op = RenderingDevice.BLEND_OP_ADD
attachment.color_blend_op = RenderingDevice.BLEND_OP_ADD
attachment.src_color_blend_factor =
RenderingDevice.BLEND_FACTOR_SRC_ALPHA
attachment.dst_color_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE
attachment.src_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_SRC_ALPHA
attachment.dst_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE
```

Subtract:

```
var attachment = RDPipelineColorBlendStateAttachment.new()
attachment.enable_blend = true
attachment.alpha_blend_op =
RenderingDevice.BLEND_OP_REVERSE_SUBTRACT
attachment.color_blend_op =
RenderingDevice.BLEND_OP_REVERSE_SUBTRACT
attachment.src_color_blend_factor =
RenderingDevice.BLEND_FACTOR_SRC_ALPHA
attachment.dst_color_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE
attachment.src_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_SRC_ALPHA
attachment.dst_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE
```

Multiply:

```
var attachment = RDPipelineColorBlendStateAttachment.new()
attachment.enable_blend = true
attachment.alpha_blend_op = RenderingDevice.BLEND_OP_ADD
attachment.color_blend_op = RenderingDevice.BLEND_OP_ADD
attachment.src_color_blend_factor =
RenderingDevice.BLEND_FACTOR_DST_COLOR
attachment.dst_color_blend_factor =
RenderingDevice.BLEND_FACTOR_ZERO
attachment.src_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_DST_ALPHA
attachment.dst_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_ZERO
```

Pre-multiplied alpha:

```
var attachment = RDPipelineColorBlendStateAttachment.new()
attachment.enable_blend = true
attachment.alpha_blend_op = RenderingDevice.BLEND_OP_ADD
attachment.color_blend_op = RenderingDevice.BLEND_OP_ADD
attachment.src_color_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE
attachment.dst_color_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE_MINUS_SRC_ALPHA
attachment.src_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE
attachment.dst_alpha_blend_factor =
RenderingDevice.BLEND_FACTOR_ONE_MINUS_SRC_ALPHA
```

Properties

BlendOperation	alpha_blend_op	`0`
BlendOperation	color_blend_op	`0`
BlendFactor	dst_alpha_blend_factor	`0`
BlendFactor	dst_color_blend_factor	`0`
bool	enable_blend	`false`
BlendFactor	src_alpha_blend_factor	`0`
BlendFactor	src_color_blend_factor	`0`

bool	write_a	`true`
bool	write_b	`true`
bool	write_g	`true`
bool	write_r	`true`

Methods

void	set_as_mix()

Property Descriptions

BlendOperation **alpha_blend_op** = `0` 🔗

- void **set_alpha_blend_op**(value: BlendOperation)
- BlendOperation **get_alpha_blend_op**()

The blend mode to use for the alpha channel.

BlendOperation **color_blend_op** = `0` 🔗

- void **set_color_blend_op**(value: BlendOperation)
- BlendOperation **get_color_blend_op**()

The blend mode to use for the red/green/blue color channels.

BlendFactor **dst_alpha_blend_factor** = `0`

- void **set_dst_alpha_blend_factor**(value: BlendFactor)
- BlendFactor **get_dst_alpha_blend_factor**()

Controls how the blend factor for the alpha channel is determined based on the destination's fragments.

BlendFactor **dst_color_blend_factor** = `0`

- void **set_dst_color_blend_factor**(value: BlendFactor)
- BlendFactor **get_dst_color_blend_factor**()

Controls how the blend factor for the color channels is determined based on the destination's fragments.

bool **enable_blend** = `false`

- void **set_enable_blend**(value: bool)
- bool **get_enable_blend**()

If `true`, performs blending between the source and destination according to the factors defined in src_color_blend_factor, dst_color_blend_factor, src_alpha_blend_factor and dst_alpha_blend_factor. The blend modes color_blend_op and alpha_blend_op are also taken into account, with write_r, write_g, write_b and write_a controlling the output.

BlendFactor **src_alpha_blend_factor** = `0`

- void **set_src_alpha_blend_factor**(value: BlendFactor)

- BlendFactor **get_src_alpha_blend_factor**()

Controls how the blend factor for the alpha channel is determined based on the source's fragments.

BlendFactor **src_color_blend_factor** = `0`

- void **set_src_color_blend_factor**(value: BlendFactor)
- BlendFactor **get_src_color_blend_factor**()

Controls how the blend factor for the color channels is determined based on the source's fragments.

---

bool **write_a** = `true`

- void **set_write_a**(value: bool)
- bool **get_write_a**()

If `true`, writes the new alpha channel to the final result.

---

bool **write_b** = `true`

- void **set_write_b**(value: bool)
- bool **get_write_b**()

If `true`, writes the new blue color channel to the final result.

---

bool **write_g** = `true`

- void **set_write_g**(value: bool)
- bool **get_write_g**()

If `true`, writes the new green color channel to the final result.

---

bool **write_r** = `true`

- void **set_write_r**(value: bool)
- bool **get_write_r**()

If `true`, writes the new red color channel to the final result.

Method Descriptions

void **set_as_mix**()

Convenience method to perform standard mix blending with straight (non-premultiplied) alpha. This sets enable_blend to `true`, src_color_blend_factor to RenderingDevice.BLEND_FACTOR_SRC_ALPHA, dst_color_blend_factor to RenderingDevice.BLEND_FACTOR_ONE_MINUS_SRC_ALPHA, src_alpha_blend_factor to RenderingDevice.BLEND_FACTOR_SRC_ALPHA and dst_alpha_blend_factor to RenderingDevice.BLEND_FACTOR_ONE_MINUS_SRC_ALPHA.

RDPipelineDepthStencilState

Inherits: RefCounted **<** Object

Pipeline depth/stencil state (used by RenderingDevice).

Description

RDPipelineDepthStencilState controls the way depth and stencil comparisons are performed when sampling those values using RenderingDevice.

Properties

CompareOperator	back_op_compare	`7`
int	back_op_compare_mask	`0`
StencilOperation	back_op_depth_fail	`1`
StencilOperation	back_op_fail	`1`
StencilOperation	back_op_pass	`1`
int	back_op_reference	`0`

int	back_op_write_mask	`0`
CompareOperator	depth_compare_operator	`7`
float	depth_range_max	`0.0`
float	depth_range_min	`0.0`
bool	enable_depth_range	`false`
bool	enable_depth_test	`false`
bool	enable_depth_write	`false`
bool	enable_stencil	`false`
CompareOperator	front_op_compare	`7`
int	front_op_compare_mask	`0`
StencilOperation	front_op_depth_fail	`1`
StencilOperation	front_op_fail	`1`
StencilOperation	front_op_pass	`1`

int	front_op_reference	0
int	front_op_write_mask	0

Property Descriptions

CompareOperator **back_op_compare** = `7` 🔗

- void **set_back_op_compare**(value: CompareOperator)
- CompareOperator **get_back_op_compare**()

The method used for comparing the previous back stencil value and back_op_reference.

---

int **back_op_compare_mask** = `0` 🔗

- void **set_back_op_compare_mask**(value: int)
- int **get_back_op_compare_mask**()

Selects which bits from the back stencil value will be compared.

StencilOperation **back_op_depth_fail** = `1` 🔗

- void **set_back_op_depth_fail**(value: StencilOperation)
- StencilOperation **get_back_op_depth_fail**()

The operation to perform on the stencil buffer for back pixels that pass the stencil test but fail the depth test.

StencilOperation **back_op_fail** = `1`

- void **set_back_op_fail**(value: StencilOperation)
- StencilOperation **get_back_op_fail**()

The operation to perform on the stencil buffer for back pixels that fail the stencil test

---

StencilOperation **back_op_pass** = `1`

- void **set_back_op_pass**(value: StencilOperation)
- StencilOperation **get_back_op_pass**()

The operation to perform on the stencil buffer for back pixels that pass the stencil test.

---

int **back_op_reference** = `0`

- void **set_back_op_reference**(value: int)
- int **get_back_op_reference**()

The value the previous back stencil value will be compared to.

---

int **back_op_write_mask** = `0`

- void **set_back_op_write_mask**(value: int)
- int **get_back_op_write_mask**()

Selects which bits from the back stencil value will be changed.

CompareOperator **depth_compare_operator** = `7`

- void **set_depth_compare_operator**(value: CompareOperator)
- CompareOperator **get_depth_compare_operator**()

The method used for comparing the previous and current depth values.

---

float **depth_range_max** = `0.0` 🔗

- void **set_depth_range_max**(value: float)
- float **get_depth_range_max**()

The maximum depth that returns true for enable_depth_range.

---

float **depth_range_min** = `0.0` 🔗

- void **set_depth_range_min**(value: float)
- float **get_depth_range_min**()

The minimum depth that returns true for enable_depth_range.

---

bool **enable_depth_range** = `false` 🔗

- void **set_enable_depth_range**(value: bool)
- bool **get_enable_depth_range**()

If `true`, each depth value will be tested to see if it is between depth_range_min and depth_range_max. If it is outside of these values, it is discarded.

---

bool **enable_depth_test** = `false` 🔗

- void **set_enable_depth_test**(value: bool)

- bool **get_enable_depth_test**()

If `true`, enables depth testing which allows objects to be automatically occluded by other objects based on their depth. This also allows objects to be partially occluded by other objects. If `false`, objects will appear in the order they were drawn (like in Godot's 2D renderer).

---

bool **enable_depth_write** = `false` 🔗

- void **set_enable_depth_write**(value: bool)
- bool **get_enable_depth_write**()

If `true`, writes to the depth buffer whenever the depth test returns true. Only works when enable_depth_test is also true.

---

bool **enable_stencil** = `false` 🔗

- void **set_enable_stencil**(value: bool)
- bool **get_enable_stencil**()

If `true`, enables stencil testing. There are separate stencil buffers for front-facing triangles and back-facing triangles. See properties that begin with "front_op" and properties with "back_op" for each.

CompareOperator **front_op_compare** = `7` 🔗

- void **set_front_op_compare**(value: CompareOperator)
- CompareOperator **get_front_op_compare**()

The method used for comparing the previous front stencil value and front_op_reference.

---

int **front_op_compare_mask** = `0`

- void **set_front_op_compare_mask**(value: int)
- int **get_front_op_compare_mask**()

Selects which bits from the front stencil value will be compared.

StencilOperation **front_op_depth_fail** = `1`

- void **set_front_op_depth_fail**(value: StencilOperation)
- StencilOperation **get_front_op_depth_fail**()

The operation to perform on the stencil buffer for front pixels that pass the stencil test but fail the depth test.

StencilOperation **front_op_fail** = `1`

- void **set_front_op_fail**(value: StencilOperation)
- StencilOperation **get_front_op_fail**()

The operation to perform on the stencil buffer for front pixels that fail the stencil test.

StencilOperation **front_op_pass** = `1`

- void **set_front_op_pass**(value: StencilOperation)
- StencilOperation **get_front_op_pass**()

The operation to perform on the stencil buffer for front pixels that pass the stencil test.

---

int **front_op_reference** = `0`

- void **set_front_op_reference**(value: int)
- int **get_front_op_reference**()

The value the previous front stencil value will be compared to.

int **front_op_write_mask** = 0 🔗

- void **set_front_op_write_mask**(value: int)
- int **get_front_op_write_mask**()

Selects which bits from the front stencil value will be changed.

RDPipelineMultisampleState

Inherits: RefCounted **<** Object

Pipeline multisample state (used by RenderingDevice).

Description

RDPipelineMultisampleState is used to control how multisample or supersample antialiasing is being performed when rendering using RenderingDevice.

Properties

bool	enable_alpha_to_coverage	`false`
bool	enable_alpha_to_one	`false`
bool	enable_sample_shading	`false`
float	min_sample_shading	`0.0`
TextureSamples	sample_count	`0`
Array[int]	sample_masks	`[]`

Property Descriptions

bool **enable_alpha_to_coverage** = `false` 🔗

- void **set_enable_alpha_to_coverage**(value: bool)
- bool **get_enable_alpha_to_coverage**()

If `true`, alpha to coverage is enabled. This generates a temporary coverage value based on the alpha component of the fragment's first color output. This allows alpha transparency to make use of multisample antialiasing.

---

bool **enable_alpha_to_one** = `false` 🔗

- void **set_enable_alpha_to_one**(value: bool)
- bool **get_enable_alpha_to_one**()

If `true`, alpha is forced to either `0.0` or `1.0`. This allows hardening the edges of antialiased alpha transparencies. Only relevant if enable_alpha_to_coverage is `true`.

---

bool **enable_sample_shading** = `false` 🔗

- void **set_enable_sample_shading**(value: bool)
- bool **get_enable_sample_shading**()

If `true`, enables per-sample shading which replaces MSAA by SSAA. This provides higher quality antialiasing that works with transparent (alpha scissor) edges. This has a very high performance cost. See also min_sample_shading. See the per-sample shading Vulkan documentation [https://registry.khronos.org/vulkan/specs/1.3-extensions/html/vkspec.html#primsrast-sampleshading] for more details.

---

float **min_sample_shading** = `0.0` 🔗

- void **set_min_sample_shading**(value: float)
- float **get_min_sample_shading**()

The multiplier of sample_count that determines how many samples are performed for each fragment. Must be between `0.0` and `1.0` (inclusive). Only effective if enable_sample_shading is `true`. If min_sample_shading is `1.0`, fragment invocation must only read from the coverage index sample. Tile image access must not be used if enable_sample_shading is *not* `1.0`.

TextureSamples **sample_count** = `0` 🔗

- void **set_sample_count**(value: TextureSamples)
- TextureSamples **get_sample_count**()

The number of MSAA samples (or SSAA samples if enable_sample_shading is `true`) to perform. Higher values result in better antialiasing, at the cost of performance.

Array[int] **sample_masks** = `[]` 🔗

- void **set_sample_masks**(value: Array[int])
- Array[int] **get_sample_masks**()

The sample mask array. See the sample mask Vulkan documentation [https://registry.khronos.org/vulkan/specs/1.3-extensions/html/vkspec.html#fragops-samplemask] for more details.

RDPipelineRasterizationState

Inherits: RefCounted **<** Object

Pipeline rasterization state (used by RenderingDevice).

Description

This object is used by RenderingDevice.

Properties

PolygonCullMode	cull_mode	`0`
float	depth_bias_clamp	`0.0`
float	depth_bias_constant_factor	`0.0`
bool	depth_bias_enabled	`false`
float	depth_bias_slope_factor	`0.0`
bool	discard_primitives	`false`
bool	enable_depth_clamp	`false`

PolygonFrontFace	front_face	`0`
float	line_width	`1.0`
int	patch_control_points	`1`
bool	wireframe	`false`

Property Descriptions

PolygonCullMode **cull_mode** = `0` 🔗

- void **set_cull_mode**(value: PolygonCullMode)
- PolygonCullMode **get_cull_mode**()

The cull mode to use when drawing polygons, which determines whether front faces or backfaces are hidden.

---

float **depth_bias_clamp** = `0.0` 🔗

- void **set_depth_bias_clamp**(value: float)
- float **get_depth_bias_clamp**()

A limit for how much each depth value can be offset. If negative, it serves as a minimum value, but if positive, it serves as a maximum value.

---

float **depth_bias_constant_factor** = `0.0` 🔗

- void **set_depth_bias_constant_factor**(value: float)
- float **get_depth_bias_constant_factor**()

A constant offset added to each depth value. Applied after depth_bias_slope_factor.

---

bool **depth_bias_enabled** = `false`

- void **set_depth_bias_enabled**(value: bool)
- bool **get_depth_bias_enabled**()

If `true`, each generated depth value will by offset by some amount. The specific amount is generated per polygon based on the values of depth_bias_slope_factor and depth_bias_constant_factor.

---

float **depth_bias_slope_factor** = `0.0`

- void **set_depth_bias_slope_factor**(value: float)
- float **get_depth_bias_slope_factor**()

A constant scale applied to the slope of each polygons' depth. Applied before depth_bias_constant_factor.

---

bool **discard_primitives** = `false`

- void **set_discard_primitives**(value: bool)
- bool **get_discard_primitives**()

If `true`, primitives are discarded immediately before the rasterization stage.

---

bool **enable_depth_clamp** = `false`

- void **set_enable_depth_clamp**(value: bool)
- bool **get_enable_depth_clamp**()

If `true`, clamps depth values according to the minimum and maximum depth of the associated viewport.

PolygonFrontFace **front_face** = `0`

- void **set_front_face**(value: PolygonFrontFace)
- PolygonFrontFace **get_front_face**()

The winding order to use to determine which face of a triangle is considered its front face.

---

float **line_width** = `1.0`

- void **set_line_width**(value: float)
- float **get_line_width**()

The line width to use when drawing lines (in pixels). Thick lines may not be supported on all hardware.

---

int **patch_control_points** = `1`

- void **set_patch_control_points**(value: int)
- int **get_patch_control_points**()

The number of control points to use when drawing a patch with tessellation enabled. Higher values result in higher quality at the cost of performance.

---

bool **wireframe** = `false`

- void **set_wireframe**(value: bool)
- bool **get_wireframe**()

If `true`, performs wireframe rendering for triangles instead of flat or textured rendering.

RDPipelineSpecializationConstant

Inherits: RefCounted **<** Object

Pipeline specialization constant (used by RenderingDevice).

Description

A *specialization constant* is a way to create additional variants of shaders without actually increasing the number of shader versions that are compiled. This allows improving performance by reducing the number of shader versions and reducing `if` branching, while still allowing shaders to be flexible for different use cases.

This object is used by RenderingDevice.

Properties

int	constant_id	`0`
Variant	value	

Property Descriptions

int **constant_id** = `0` 🔗

- void **set_constant_id**(value: int)
- int **get_constant_id**()

The identifier of the specialization constant. This is a value starting from `0` and that increments for every different specialization constant for a given shader.

Variant **value** 🔗

- void **set_value**(value: Variant)
- Variant **get_value**()

The specialization constant's value. Only bool, int and float types are valid for specialization constants.

RDSamplerState

Inherits: RefCounted **<** Object

Sampler state (used by RenderingDevice).

Description

This object is used by RenderingDevice.

Properties

float	anisotropy_max	`1.0`
SamplerBorderColor	border_color	`2`
CompareOperator	compare_op	`7`
bool	enable_compare	`false`
float	lod_bias	`0.0`
SamplerFilter	mag_filter	`0`
float	max_lod	`1e+20`

SamplerFilter	min_filter	`0`
float	min_lod	`0.0`
SamplerFilter	mip_filter	`0`
SamplerRepeatMode	repeat_u	`2`
SamplerRepeatMode	repeat_v	`2`
SamplerRepeatMode	repeat_w	`2`
bool	unnormalized_uvw	`false`
bool	use_anisotropy	`false`

Property Descriptions

float **anisotropy_max** = `1.0` 🔗

- void **set_anisotropy_max**(value: float)
- float **get_anisotropy_max**()

Maximum anisotropy that can be used when sampling. Only effective if use_anisotropy is `true`. Higher values result in a sharper sampler at oblique angles, at the cost of performance (due to memory bandwidth). This value may be limited by the graphics hardware in use. Most graphics hardware only supports values up to `16.0`.

If anisotropy_max is `1.0`, forcibly disables anisotropy even if use_anisotropy is `true`.

SamplerBorderColor **border_color** = `2` 🔗

- void **set_border_color**(value: SamplerBorderColor)
- SamplerBorderColor **get_border_color**()

The border color that will be returned when sampling outside the sampler's bounds and the repeat_u, repeat_v or repeat_w modes have repeating disabled.

CompareOperator **compare_op** = `7` 🔗

- void **set_compare_op**(value: CompareOperator)
- CompareOperator **get_compare_op**()

The compare operation to use. Only effective if enable_compare is `true`.

bool **enable_compare** = `false` 🔗

- void **set_enable_compare**(value: bool)
- bool **get_enable_compare**()

If `true`, returned values will be based on the comparison operation defined in compare_op. This is a hardware-based approach and is therefore faster than performing this manually in a shader. For example, compare operations are used for shadow map rendering by comparing depth values from a shadow sampler.

float **lod_bias** = `0.0`

- void **set_lod_bias**(value: float)
- float **get_lod_bias**()

The mipmap LOD bias to use. Positive values will make the sampler blurrier at a given distance, while negative values will make the sampler sharper at a given distance (at the risk of looking grainy). Recommended values are between `-0.5` and `0.0`. Only effective if the sampler has mipmaps available.

SamplerFilter **mag_filter** = `0`

- void **set_mag_filter**(value: SamplerFilter)
- SamplerFilter **get_mag_filter**()

The sampler's magnification filter. It is the filtering method used when sampling texels that appear bigger than on-screen pixels.

float **max_lod** = `1e+20`

- void **set_max_lod**(value: float)
- float **get_max_lod**()

The maximum mipmap LOD bias to display (lowest resolution). Only effective if the sampler has mipmaps available.

SamplerFilter **min_filter** = `0`

- void **set_min_filter**(value: SamplerFilter)
- SamplerFilter **get_min_filter**()

The sampler's minification filter. It is the filtering method used when sampling texels that appear smaller than on-screen pixels.

---

float **min_lod** = `0.0`

- void **set_min_lod**(value: float)
- float **get_min_lod**()

The minimum mipmap LOD bias to display (highest resolution). Only effective if the sampler has mipmaps available.

SamplerFilter **mip_filter** = `0`

- void **set_mip_filter**(value: SamplerFilter)
- SamplerFilter **get_mip_filter**()

The filtering method to use for mipmaps.

SamplerRepeatMode **repeat_u** = `2`

- void **set_repeat_u**(value: SamplerRepeatMode)
- SamplerRepeatMode **get_repeat_u**()

The repeat mode to use along the U axis of UV coordinates. This affects the returned values if sampling outside the UV bounds.

SamplerRepeatMode **repeat_v** = `2`

- void **set_repeat_v**(value: SamplerRepeatMode)
- SamplerRepeatMode **get_repeat_v**()

The repeat mode to use along the V axis of UV coordinates. This affects the returned values if sampling outside the UV bounds.

SamplerRepeatMode **repeat_w** = `2`

- void **set_repeat_w**(value: SamplerRepeatMode)
- SamplerRepeatMode **get_repeat_w**()

The repeat mode to use along the W axis of UV coordinates. This affects the returned values if sampling outside the UV bounds. Only effective for 3D samplers.

---

bool **unnormalized_uvw** = `false` 🔗

- void **set_unnormalized_uvw**(value: bool)
- bool **get_unnormalized_uvw**()

If `true`, the texture will be sampled with coordinates ranging from 0 to the texture's resolution. Otherwise, the coordinates will be normalized and range from 0 to 1.

---

bool **use_anisotropy** = `false` 🔗

- void **set_use_anisotropy**(value: bool)
- bool **get_use_anisotropy**()

If `true`, perform anisotropic sampling. See anisotropy_max.

RDShaderSource

Inherits: RefCounted **<** Object

Shader source code (used by RenderingDevice).

Description

Shader source code in text form.

See also RDShaderFile. **RDShaderSource** is only meant to be used with the RenderingDevice API. It should not be confused with Godot's own Shader resource, which is what Godot's various nodes use for high-level shader programming.

Properties

ShaderLanguage	language	0
String	source_compute	""
String	source_fragment	""
String	source_tesselation_control	""
String	source_tesselation_evaluation	""

String	source_vertex	""

Methods

String	get_stage_source(stage: ShaderStage) const
void	set_stage_source(stage: ShaderStage, source: String)

Property Descriptions

ShaderLanguage **language** = `0` 🔗

- void **set_language**(value: ShaderLanguage)
- ShaderLanguage **get_language**()

The language the shader is written in.

String **source_compute** = `""` 🔗

- void **set_stage_source**(stage: ShaderStage, source: String)
- String **get_stage_source**(stage: ShaderStage) const

Source code for the shader's compute stage.

String **source_fragment** = `""` 🔗

- void **set_stage_source**(stage: ShaderStage, source: String)

- String **get_stage_source**(stage: ShaderStage) const

Source code for the shader's fragment stage.

String **source_tesselation_control** = `""`

- void **set_stage_source**(stage: ShaderStage, source: String)
- String **get_stage_source**(stage: ShaderStage) const

Source code for the shader's tessellation control stage.

String **source_tesselation_evaluation** = `""`

- void **set_stage_source**(stage: ShaderStage, source: String)
- String **get_stage_source**(stage: ShaderStage) const

Source code for the shader's tessellation evaluation stage.

String **source_vertex** = `""`

- void **set_stage_source**(stage: ShaderStage, source: String)
- String **get_stage_source**(stage: ShaderStage) const

Source code for the shader's vertex stage.

Method Descriptions

String **get_stage_source**(stage: ShaderStage) const

Returns source code for the specified shader `stage`. Equivalent to getting one of source_compute, source_fragment, source_tesselation_control, source_tesselation_evaluation or source_vertex.

---

void **set_stage_source**(stage: ShaderStage, source: String) 🔗

Sets `source` code for the specified shader `stage`. Equivalent to setting one of source_compute, source_fragment, source_tesselation_control, source_tesselation_evaluation or source_vertex.

RDTextureFormat

Inherits: RefCounted **<** Object

Texture format (used by RenderingDevice).

Description

This object is used by RenderingDevice.

Properties

int	array_layers	`1`
int	depth	`1`
DataFormat	format	`8`
int	height	`1`
int	mipmaps	`1`
TextureSamples	samples	`0`
TextureType	texture_type	`1`

BitField[TextureUsageBits]	usage_bits	`0`
int	width	`1`

Methods

void	add_shareable_format(format: DataFormat)
void	remove_shareable_format(format: DataFormat)

Property Descriptions

int **array_layers** = `1` 🔗

- void **set_array_layers**(value: int)
- int **get_array_layers**()

The number of layers in the texture. Only relevant for 2D texture arrays.

---

int **depth** = `1` 🔗

- void **set_depth**(value: int)
- int **get_depth**()

The texture's depth (in pixels). This is always `1` for 2D textures.

DataFormat **format** = `8`

- void **set_format**(value: DataFormat)
- DataFormat **get_format**()

The texture's pixel data format.

---

int **height** = `1`

- void **set_height**(value: int)
- int **get_height**()

The texture's height (in pixels).

---

int **mipmaps** = `1`

- void **set_mipmaps**(value: int)
- int **get_mipmaps**()

The number of mipmaps available in the texture.

TextureSamples **samples** = `0`

- void **set_samples**(value: TextureSamples)
- TextureSamples **get_samples**()

The number of samples used when sampling the texture.

TextureType **texture_type** = `1`

- void **set_texture_type**(value: TextureType)
- TextureType **get_texture_type**()

The texture type.

BitField[TextureUsageBits] **usage_bits** = `0`

- void **set_usage_bits**(value: BitField[TextureUsageBits])
- BitField[TextureUsageBits] **get_usage_bits**()

The texture's usage bits, which determine what can be done using the texture.

---

int **width** = `1`

- void **set_width**(value: int)
- int **get_width**()

The texture's width (in pixels).

Method Descriptions

void **add_shareable_format**(format: DataFormat)

There is currently no description for this method. Please help us by contributing one!

---

void **remove_shareable_format**(format: DataFormat)

There is currently no description for this method. Please help us by contributing one!

RDTextureView

Inherits: RefCounted **<** Object

Texture view (used by RenderingDevice).

Description

This object is used by RenderingDevice.

Properties

DataFormat	format_override	`218`
TextureSwizzle	swizzle_a	`6`
TextureSwizzle	swizzle_b	`5`
TextureSwizzle	swizzle_g	`4`
TextureSwizzle	swizzle_r	`3`

Property Descriptions

DataFormat **format_override** = `218` 🔗

- void **set_format_override**(value: DataFormat)
- DataFormat **get_format_override**()

Optional override for the data format to return sampled values in. The default value of RenderingDevice.DATA_FORMAT_MAX does not override the format.

TextureSwizzle **swizzle_a** = `6` 🔗

- void **set_swizzle_a**(value: TextureSwizzle)
- TextureSwizzle **get_swizzle_a**()

The channel to sample when sampling the alpha channel.

TextureSwizzle **swizzle_b** = `5` 🔗

- void **set_swizzle_b**(value: TextureSwizzle)
- TextureSwizzle **get_swizzle_b**()

The channel to sample when sampling the blue color channel.

TextureSwizzle **swizzle_g** = `4` 🔗

- void **set_swizzle_g**(value: TextureSwizzle)
- TextureSwizzle **get_swizzle_g**()

The channel to sample when sampling the green color channel.

TextureSwizzle **swizzle_r** = `3` 🔗

- void **set_swizzle_r**(value: TextureSwizzle)
- TextureSwizzle **get_swizzle_r**()

The channel to sample when sampling the red color channel.

RDUniform

Inherits: RefCounted **<** Object

Shader uniform (used by RenderingDevice).

Description

This object is used by RenderingDevice.

Properties

int	binding	0
UniformType	uniform_type	3

Methods

void	add_id(id: RID)
void	clear_ids()
Array[RID]	get_ids() const

Property Descriptions

int **binding** = `0`

- void **set_binding**(value: int)
- int **get_binding**()

The uniform's binding.

UniformType **uniform_type** = `3`

- void **set_uniform_type**(value: UniformType)
- UniformType **get_uniform_type**()

The uniform's data type.

Method Descriptions

void **add_id**(id: RID)

Binds the given id to the uniform. The data associated with the id is then used when the uniform is passed to a shader.

---

void **clear_ids**()

Unbinds all ids currently bound to the uniform.

Array[RID] **get_ids**() const

Returns an array of all ids currently bound to the uniform.

RDVertexAttribute

Inherits: RefCounted **<** Object

Vertex attribute (used by RenderingDevice).

Description

This object is used by RenderingDevice.

Properties

DataFormat	format	`218`
VertexFrequency	frequency	`0`
int	location	`0`
int	offset	`0`
int	stride	`0`

Property Descriptions

DataFormat **format** = `218` 🔗

- void **set_format**(value: DataFormat)
- DataFormat **get_format**()

There is currently no description for this property. Please help us by contributing one!

VertexFrequency **frequency** = 0

- void **set_frequency**(value: VertexFrequency)
- VertexFrequency **get_frequency**()

There is currently no description for this property. Please help us by contributing one!

---

int **location** = 0

- void **set_location**(value: int)
- int **get_location**()

There is currently no description for this property. Please help us by contributing one!

---

int **offset** = 0

- void **set_offset**(value: int)
- int **get_offset**()

There is currently no description for this property. Please help us by contributing one!

---

int **stride** = 0

- void **set_stride**(value: int)

- int **get_stride**()

There is currently no description for this property. Please help us by contributing one!

RefCounted

Inherits: Object

Inherited By: AESContext, AStar2D, AStar3D, AStarGrid2D, AudioEffectInstance, AudioSample, AudioSamplePlayback, AudioStreamPlayback, CameraFeed, CharFXTransform, ConfigFile, Crypto, DirAccess, DTLSServer, EditorDebuggerPlugin, EditorDebuggerSession, EditorExportPlatform, EditorExportPlugin, EditorFeatureProfile, EditorFileSystemImportFormatSupportQuery, EditorInspectorPlugin, EditorResourceConversionPlugin, EditorResourcePreviewGenerator, EditorResourceTooltipPlugin, EditorSceneFormatImporter, EditorScenePostImport, EditorScenePostImportPlugin, EditorScript, EditorTranslationParserPlugin, EncodedObjectAsID, ENetConnection, EngineProfiler, Expression, FileAccess, HashingContext, HMACContext, HTTPClient, ImageFormatLoader, JavaClass, JavaScriptObject, KinematicCollision2D, KinematicCollision3D, Lightmapper, MeshConvexDecompositionSettings, MeshDataTool, MultiplayerAPI, Mutex, NavigationPathQueryParameters2D, NavigationPathQueryParameters3D, NavigationPathQueryResult2D, NavigationPathQueryResult3D, Node3DGizmo, OggPacketSequencePlayback, OpenXRAPIExtension, PackedDataContainerRef, PacketPeer, PCKPacker, PhysicsPointQueryParameters2D, PhysicsPointQueryParameters3D, PhysicsRayQueryParameters2D, PhysicsRayQueryParameters3D, PhysicsShapeQueryParameters2D, PhysicsShapeQueryParameters3D, PhysicsTestMotionParameters2D, PhysicsTestMotionParameters3D, PhysicsTestMotionResult2D, PhysicsTestMotionResult3D, RandomNumberGenerator, RDAttachmentFormat, RDFramebufferPass, RDPipelineColorBlendState, RDPipelineColorBlendStateAttachment, RDPipelineDepthStencilState, RDPipelineMultisampleState, RDPipelineRasterizationState, RDPipelineSpecializationConstant, RDSamplerState, RDShaderSource, RDTextureFormat, RDTextureView, RDUniform, RDVertexAttribute, RegEx, RegExMatch, RenderSceneBuffers, RenderSceneBuffersConfiguration, Resource, ResourceFormatLoader, ResourceFormatSaver, ResourceImporter, SceneState, SceneTreeTimer, Semaphore, SkinReference, StreamPeer, SurfaceTool, TCPServer, TextLine, TextParagraph, TextServer, Thread, TLSOptions, TriangleMesh, Tween, Tweener, UDPServer, UPNP, UPNPDevice, WeakRef, WebRTCPeerConnection, XMLParser, XRInterface, XRPose, XRTracker, ZIPPacker, ZIPReader

Base class for reference-counted objects.

Description

Base class for any object that keeps a reference count. Resource and many other helper objects inherit this class.

Unlike other Object types, **RefCounted**s keep an internal reference counter so that they are automatically released when no longer in use, and only then. **RefCounted**s therefore do not need to be freed manually with Object.free.

RefCounted instances caught in a cyclic reference will **not** be freed automatically. For example, if a node holds a reference to instance `A`, which directly or indirectly holds a reference back to `A`, `A`'s reference count will be 2. Destruction of the node will leave `A` dangling with a reference count of 1, and there will be a memory leak. To prevent this, one of the references in the cycle can be made weak with @GlobalScope.weakref.

In the vast majority of use cases, instantiating and using **RefCounted**-derived types is all you need to do. The methods provided in this class are only for advanced users, and can cause issues if misused.

Note: In C#, reference-counted objects will not be freed instantly after they are no longer in use. Instead, garbage collection will run periodically and will free reference-counted objects that are no longer in use. This means that unused ones will remain in memory for a while before being removed.

Tutorials

- When and how to avoid using nodes for everything

Methods

int	get_reference_count() const
bool	init_ref()
bool	reference()
bool	unreference()

Method Descriptions

int **get_reference_count**() const 🔗

Returns the current reference count.

---

bool **init_ref**() 🔗

Initializes the internal reference counter. Use this only if you really know what you are doing.

Returns whether the initialization was successful.

bool **reference**() 🔗

Increments the internal reference counter. Use this only if you really know what you are doing.

Returns `true` if the increment was successful, `false` otherwise.

bool **unreference**() 🔗

Decrements the internal reference counter. Use this only if you really know what you are doing.

Returns `true` if the object should be freed after the decrement, `false` otherwise.

RegEx

Inherits: RefCounted **<** Object

Class for searching text for patterns using regular expressions.

Description

A regular expression (or regex) is a compact language that can be used to recognize strings that follow a specific pattern, such as URLs, email addresses, complete sentences, etc. For example, a regex of `ab[0-9]` would find any string that is `ab` followed by any number from `0` to `9`. For a more in-depth look, you can easily find various tutorials and detailed explanations on the Internet.

To begin, the RegEx object needs to be compiled with the search pattern using compile before it can be used.

```
var regex = RegEx.new()
regex.compile("\\w-(\\d+)")
```

The search pattern must be escaped first for GDScript before it is escaped for the expression. For example, `compile("\\d+")` would be read by RegEx as `\d+`. Similarly, `compile("\"(?:\\\\.|[^\"])*\"")` would be read as `"(?:\\.|[^"])*"`. In GDScript, you can also use raw string literals (r-strings). For example, `compile(r'"(?:\\.|[^"])*"')` would be read the same.

Using search, you can find the pattern within the given text. If a pattern is found, RegExMatch is returned and you can retrieve details of the results using methods such as RegExMatch.get_string and RegExMatch.get_start.

```
var regex = RegEx.new()
regex.compile("\\w-(\\d+)")
var result = regex.search("abc n-0123")
if result:
    print(result.get_string()) # Would print n-0123
```

The results of capturing groups `()` can be retrieved by passing the group number to the various methods in RegExMatch. Group 0 is the default and will always refer to the entire pattern. In the above example, calling `result.get_string(1)` would give you `0123`.

This version of RegEx also supports named capturing groups, and the names can be used to retrieve the results. If two or more groups have the same name, the name would only refer to the first one with a match.

```
var regex = RegEx.new()
regex.compile("d(?<digit>[0-9]+)|x(?<digit>[0-9a-f]+)")
var result = regex.search("the number is x2f")
if result:
    print(result.get_string("digit")) # Would print 2f
```

If you need to process multiple results, search_all generates a list of all non-overlapping results. This can be combined with a `for` loop for convenience.

```
for result in regex.search_all("d01, d03, d0c, x3f and x42"):
    print(result.get_string("digit"))
# Would print 01 03 0 3f 42
```

Example of splitting a string using a RegEx:

```
var regex = RegEx.new()
regex.compile("\\S+") # Negated whitespace character class.
var results = []
for result in regex.search_all("One  Two \n\tThree"):
    results.push_back(result.get_string())
# The `results` array now contains "One", "Two", "Three".
```

Note: Godot's regex implementation is based on the PCRE2 [https://www.pcre.org/] library. You can view the full pattern reference here [https://www.pcre.org/current/doc/html/pcre2pattern.html].

Tip: You can use Regexr [https://regexr.com/] to test regular expressions online.

Methods

void	clear()
Error	compile(pattern: String)
RegEx	create_from_string(pattern: String) static
int	get_group_count() const
PackedStringArray	get_names() const
String	get_pattern() const
bool	is_valid() const
RegExMatch	search(subject: String, offset: int = 0, end: int = -1) const

Array[RegExMatch]	search_all(subject: String, offset: int = 0, end: int = -1) const
String	sub(subject: String, replacement: String, all: bool = false, offset: int = 0, end: int = -1) const

Method Descriptions

void **clear**()

This method resets the state of the object, as if it was freshly created. Namely, it unassigns the regular expression of this object.

Error **compile**(pattern: String)

Compiles and assign the search pattern to use. Returns @GlobalScope.OK if the compilation is successful. If an error is encountered, details are printed to standard output and an error is returned.

RegEx **create_from_string**(pattern: String) static

Creates and compiles a new **RegEx** object.

---

int **get_group_count**() const

Returns the number of capturing groups in compiled pattern.

PackedStringArray **get_names**() const

Returns an array of names of named capturing groups in the compiled pattern. They are ordered by appearance.

String **get_pattern**() const

Returns the original search pattern that was compiled.

---

bool **is_valid**() const

Returns whether this object has a valid search pattern assigned.

RegExMatch **search**(subject: String, offset: int = 0, end: int = -1) const

Searches the text for the compiled pattern. Returns a RegExMatch container of the first matching result if found, otherwise `null`.

The region to search within can be specified with `offset` and `end`. This is useful when searching for another match in the same `subject` by calling this method again after a previous success. Note that setting these parameters differs from passing over a shortened string. For example, the start anchor `^` is not affected by `offset`, and the character before `offset` will be checked for the word boundary `\b`.

Array[RegExMatch] **search_all**(subject: String, offset: int = 0, end: int = -1) const

Searches the text for the compiled pattern. Returns an array of RegExMatch containers for each non-overlapping result. If no results were found, an empty array is returned instead.

The region to search within can be specified with `offset` and `end`. This is useful when searching for another match in the same `subject` by calling this method again after a previous success. Note that setting these parameters differs from passing over a shortened string. For example, the start anchor `^` is not affected by `offset`, and the character before `offset` will be checked for the word boundary `\b`.

String **sub**(subject: String, replacement: String, all: bool = false, offset: int = 0, end: int = -1) const

Searches the text for the compiled pattern and replaces it with the specified string. Escapes and backreferences such as `$1` and `$name` are expanded and resolved. By default, only the first instance is replaced, but it can be changed for all instances (global replacement).

The region to search within can be specified with `offset` and `end`. This is useful when searching for another match in the same `subject` by calling this method again after a previous success. Note that setting these parameters differs from passing over a shortened string. For example, the start anchor `^` is not affected by `offset`, and the character before `offset` will be checked for the word boundary `\b`.

RegExMatch

Inherits: RefCounted **<** Object

Contains the results of a RegEx search.

Description

Contains the results of a single RegEx match returned by RegEx.search and RegEx.search_all. It can be used to find the position and range of the match and its capturing groups, and it can extract its substring for you.

Properties

Dictionary	names	`{}`
PackedStringArray	strings	`PackedStringArray()`
String	subject	`""`

Methods

int	get_end(name: Variant = 0) const

int	get_group_count() const
int	get_start(name: Variant = 0) const
String	get_string(name: Variant = 0) const

Property Descriptions

Dictionary **names** = `{}`

- Dictionary **get_names**()

A dictionary of named groups and its corresponding group number. Only groups that were matched are included. If multiple groups have the same name, that name would refer to the first matching one.

PackedStringArray **strings** = `PackedStringArray()`

- PackedStringArray **get_strings**()

An Array of the match and its capturing groups.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedStringArray for more details.

String **subject** = `""`

- String **get_subject**()

The source string used with the search pattern to find this matching result.

Method Descriptions

int **get_end**(name: Variant = 0) const

Returns the end position of the match within the source string. The end position of capturing groups can be retrieved by providing its group number as an integer or its string name (if it's a named group). The default value of 0 refers to the whole pattern.

Returns -1 if the group did not match or doesn't exist.

---

int **get_group_count**() const

Returns the number of capturing groups.

---

int **get_start**(name: Variant = 0) const

Returns the starting position of the match within the source string. The starting position of capturing groups can be retrieved by providing its group number as an integer or its string name (if it's a named group). The default value of 0 refers to the whole pattern.

Returns -1 if the group did not match or doesn't exist.

String **get_string**(name: Variant = 0) const

Returns the substring of the match from the source string. Capturing groups can be retrieved by providing its group number as an integer or its string name (if it's a named group). The default value of 0 refers to the whole pattern.

Returns an empty string if the group did not match or doesn't exist.

RenderData

Inherits: Object

Inherited By: RenderDataExtension, RenderDataRD

Abstract render data object, holds frame data related to rendering a single frame of a viewport.

Description

Abstract render data object, exists for the duration of rendering a single viewport.

Note: This is an internal rendering server object, do not instantiate this from script.

Methods

RID	get_camera_attributes() const
RID	get_environment() const
RenderSceneBuffers	get_render_scene_buffers() const
RenderSceneData	get_render_scene_data() const

Method Descriptions

RID **get_camera_attributes**() const

Returns the RID of the camera attributes object in the RenderingServer being used to render this viewport.

RID **get_environment**() const

Returns the RID of the environments object in the RenderingServer being used to render this viewport.

RenderSceneBuffers **get_render_scene_buffers**() const

Returns the RenderSceneBuffers object managing the scene buffers for rendering this viewport.

RenderSceneData **get_render_scene_data**() const

Returns the RenderSceneData object managing this frames scene data.

RenderDataExtension

Inherits: RenderData **<** Object

This class allows for a RenderData implementation to be made in GDExtension.

Description

This class allows for a RenderData implementation to be made in GDExtension.

Methods

RID	_get_camera_attributes() virtual const
RID	_get_environment() virtual const
RenderSceneBuffers	_get_render_scene_buffers() virtual const
RenderSceneData	_get_render_scene_data() virtual const

Method Descriptions

RID **_get_camera_attributes**() virtual const 🔗

Implement this in GDExtension to return the RID for the implementation's camera attributes object.

RID **_get_environment**() virtual const

Implement this in GDExtension to return the RID of the implementation's environment object.

RenderSceneBuffers **_get_render_scene_buffers**() virtual const

Implement this in GDExtension to return the implementation's RenderSceneBuffers object.

RenderSceneData **_get_render_scene_data**() virtual const

Implement this in GDExtension to return the implementation's RenderSceneDataExtension object.

RenderDataRD

Inherits: RenderData **<** Object

Render data implementation for the RenderingDevice based renderers.

Note: This is an internal rendering server object, do not instantiate this from script.

Description

This object manages all render data for the rendering device based renderers.

Note: This is an internal rendering server object only exposed for GDExtension plugins.

RenderingDevice

Inherits: Object

Abstraction for working with modern low-level graphics APIs.

Description

RenderingDevice is an abstraction for working with modern low-level graphics APIs such as Vulkan. Compared to RenderingServer (which works with Godot's own rendering subsystems), **RenderingDevice** is much lower-level and allows working more directly with the underlying graphics APIs. **RenderingDevice** is used in Godot to provide support for several modern low-level graphics APIs while reducing the amount of code duplication required. **RenderingDevice** can also be used in your own projects to perform things that are not exposed by RenderingServer or high-level nodes, such as using compute shaders.

On startup, Godot creates a global **RenderingDevice** which can be retrieved using RenderingServer.get_rendering_device. This global **RenderingDevice** performs drawing to the screen.

Local RenderingDevices: Using RenderingServer.create_local_rendering_device, you can create "secondary" rendering devices to perform drawing and GPU compute operations on separate threads.

Note: RenderingDevice assumes intermediate knowledge of modern graphics APIs such as Vulkan, Direct3D 12, Metal or WebGPU. These graphics APIs are lower-level than OpenGL or Direct3D 11, requiring you to perform what was previously done by the graphics driver itself. If you have difficulty understanding the concepts used in this class, follow the Vulkan Tutorial [https://vulkan-tutorial.com/] or Vulkan Guide [https://vkguide.dev/]. It's recommended to have existing modern OpenGL or Direct3D 11 knowledge before attempting to learn a low-level graphics API.

Note: RenderingDevice is not available when running in headless mode or when using the Compatibility rendering method.

Tutorials

- Using compute shaders

Methods

void	barrier(from: BitField[BarrierMask] = 32767, to: BitField[BarrierMask] = 32767)
Error	buffer_clear(buffer: RID, offset: int, size_bytes: int)
Error	buffer_copy(src_buffer: RID, dst_buffer: RID, src_offset: int, dst_offset: int, size: int)
PackedByteArray	buffer_get_data(buffer: RID, offset_bytes: int = 0, size_bytes: int = 0)
Error	buffer_update(buffer: RID, offset: int, size_bytes: int, data: PackedByteArray)
void	capture_timestamp(name: String)

void	compute_list_add_barrier(compute_list: int)
int	compute_list_begin()
void	compute_list_bind_compute_pipeline(compute_list: int, compute_pipeline: RID)
void	compute_list_bind_uniform_set(compute_list: int, uniform_set: RID, set_index: int)
void	compute_list_dispatch(compute_list: int, x_groups: int, y_groups: int, z_groups: int)
void	compute_list_dispatch_indirect(compute_list: int, buffer: RID, offset: int)
void	compute_list_end()
void	compute_list_set_push_constant(compute_list: int, buffer: PackedByteArray, size_bytes: int)
RID	compute_pipeline_create(shader: RID, specialization_constants: Array[RDPipelineSpecializationConstant] = [])
bool	compute_pipeline_is_valid(compute_pipeline: RID)
RenderingDevice	create_local_device()

void	draw_command_begin_label(name: String, color: Color)
void	draw_command_end_label()
void	draw_command_insert_label(name: String, color: Color)
int	draw_list_begin(framebuffer: RID, initial_color_action: InitialAction, final_color_action: FinalAction, initial_depth_action: InitialAction, final_depth_action: FinalAction, clear_color_values: PackedColorArray = PackedColorArray(), clear_depth: float = 1.0, clear_stencil: int = 0, region: Rect2 = Rect2(0, 0, 0, 0))
int	draw_list_begin_for_screen(screen: int = 0, clear_color: Color = Color(0, 0, 0, 1))
PackedInt64Array	draw_list_begin_split(framebuffer: RID, splits: int, initial_color_action: InitialAction, final_color_action: FinalAction, initial_depth_action: InitialAction, final_depth_action: FinalAction, clear_color_values: PackedColorArray = PackedColorArray(), clear_depth: float = 1.0, clear_stencil: int = 0, region: Rect2 = Rect2(0, 0, 0, 0), storage_textures: Array[RID] = [])

void	draw_list_bind_index_array(draw_list: int, index_array: RID)
void	draw_list_bind_render_pipeline(draw_list: int, render_pipeline: RID)
void	draw_list_bind_uniform_set(draw_list: int, uniform_set: RID, set_index: int)
void	draw_list_bind_vertex_array(draw_list: int, vertex_array: RID)
void	draw_list_disable_scissor(draw_list: int)
void	draw_list_draw(draw_list: int, use_indices: bool, instances: int, procedural_vertex_count: int = 0)
void	draw_list_enable_scissor(draw_list: int, rect: Rect2 = Rect2(0, 0, 0, 0))
void	draw_list_end()
void	draw_list_set_blend_constants(draw_list: int, color: Color)
void	draw_list_set_push_constant(draw_list: int, buffer: PackedByteArray, size_bytes: int)

int	draw_list_switch_to_next_pass()
PackedInt64Array	draw_list_switch_to_next_pass_split(splits: int)
RID	framebuffer_create(textures: Array[RID], validate_with_format: int = -1, view_count: int = 1)
RID	framebuffer_create_empty(size: Vector2i, samples: TextureSamples = 0, validate_with_format: int = -1)
RID	framebuffer_create_multipass(textures: Array[RID], passes: Array[RDFramebufferPass], validate_with_format: int = -1, view_count: int = 1)
int	framebuffer_format_create(attachments: Array[RDAttachmentFormat], view_count: int = 1)
int	framebuffer_format_create_empty(samples: TextureSamples = 0)
int	framebuffer_format_create_multipass(attachments: Array[RDAttachmentFormat], passes: Array[RDFramebufferPass], view_count: int = 1)
TextureSamples	framebuffer_format_get_texture_samples(format: int, render_pass: int = 0)

int	framebuffer_get_format(framebuffer: RID)
bool	framebuffer_is_valid(framebuffer: RID) const
void	free_rid(rid: RID)
void	full_barrier()
int	get_captured_timestamp_cpu_time(index: int) const
int	get_captured_timestamp_gpu_time(index: int) const
String	get_captured_timestamp_name(index: int) const
int	get_captured_timestamps_count() const
int	get_captured_timestamps_frame() const
String	get_device_name() const
String	get_device_pipeline_cache_uuid() const
String	get_device_vendor_name() const

int	get_driver_resource(resource: DriverResource, rid: RID, index: int)
int	get_frame_delay() const
int	get_memory_usage(type: MemoryType) const
RID	index_array_create(index_buffer: RID, index_offset: int, index_count: int)
RID	index_buffer_create(size_indices: int, format: IndexBufferFormat, data: PackedByteArray = PackedByteArray(), use_restart_indices: bool = false)
int	limit_get(limit: Limit) const
RID	render_pipeline_create(shader: RID, framebuffer_format: int, vertex_format: int, primitive: RenderPrimitive, rasterization_state: RDPipelineRasterizationState, multisample_state: RDPipelineMultisampleState, stencil_state: RDPipelineDepthStencilState, color_blend_state: RDPipelineColorBlendState, dynamic_state_flags: BitField[PipelineDynamicStateFlags] = 0, for_render_pass: int = 0, specialization_constants: Array[RDPipelineSpecializationConstant] = [])
bool	render_pipeline_is_valid(render_pipeline: RID)

RID	sampler_create(state: RDSamplerState)
bool	sampler_is_format_supported_for_filter(format: DataFormat, sampler_filter: SamplerFilter) const
int	screen_get_framebuffer_format(screen: int = 0) const
int	screen_get_height(screen: int = 0) const
int	screen_get_width(screen: int = 0) const
void	set_resource_name(id: RID, name: String)
PackedByteArray	shader_compile_binary_from_spirv(spirv_data: RDShaderSPIRV, name: String = "")
RDShaderSPIRV	shader_compile_spirv_from_source(shader_source: RDShaderSource, allow_cache: bool = true)
RID	shader_create_from_bytecode(binary_data: PackedByteArray, placeholder_rid: RID = RID())
RID	shader_create_from_spirv(spirv_data: RDShaderSPIRV, name: String = "")
RID	shader_create_placeholder()

int	shader_get_vertex_input_attribute_mask(shader: RID)
RID	storage_buffer_create(size_bytes: int, data: PackedByteArray = PackedByteArray(), usage: BitField[StorageBufferUsage] = 0)
void	submit()
void	sync()
RID	texture_buffer_create(size_bytes: int, format: DataFormat, data: PackedByteArray = PackedByteArray())
Error	texture_clear(texture: RID, color: Color, base_mipmap: int, mipmap_count: int, base_layer: int, layer_count: int)
Error	texture_copy(from_texture: RID, to_texture: RID, from_pos: Vector3, to_pos: Vector3, size: Vector3, src_mipmap: int, dst_mipmap: int, src_layer: int, dst_layer: int)
RID	texture_create(format: RDTextureFormat, view: RDTextureView, data: Array[PackedByteArray] = [])
RID	texture_create_from_extension(type: TextureType, format: DataFormat, samples: TextureSamples,

	usage_flags: BitField[TextureUsageBits], image: int, width: int, height: int, depth: int, layers: int)
RID	texture_create_shared(view: RDTextureView, with_texture: RID)
RID	texture_create_shared_from_slice(view: RDTextureView, with_texture: RID, layer: int, mipmap: int, mipmaps: int = 1, slice_type: TextureSliceType = 0)
PackedByteArray	texture_get_data(texture: RID, layer: int)
RDTextureFormat	texture_get_format(texture: RID)
int	texture_get_native_handle(texture: RID)
bool	texture_is_format_supported_for_usage(format: DataFormat, usage_flags: BitField[TextureUsageBits]) const
bool	texture_is_shared(texture: RID)
bool	texture_is_valid(texture: RID)
Error	texture_resolve_multisample(from_texture: RID, to_texture: RID)

Error	texture_update(texture: RID, layer: int, data: PackedByteArray)
RID	uniform_buffer_create(size_bytes: int, data: PackedByteArray = PackedByteArray())
RID	uniform_set_create(uniforms: Array[RDUniform], shader: RID, shader_set: int)
bool	uniform_set_is_valid(uniform_set: RID)
RID	vertex_array_create(vertex_count: int, vertex_format: int, src_buffers: Array[RID], offsets: PackedInt64Array = PackedInt64Array())
RID	vertex_buffer_create(size_bytes: int, data: PackedByteArray = PackedByteArray(), use_as_storage: bool = false)
int	vertex_format_create(vertex_descriptions: Array[RDVertexAttribute])

Enumerations

enum **DeviceType**: 🔗

DeviceType **DEVICE_TYPE_OTHER** = 0

Rendering device type does not match any of the other enum values or is unknown.

DeviceType **DEVICE_TYPE_INTEGRATED_GPU** = `1`

Rendering device is an integrated GPU, which is typically *(but not always)* slower than dedicated GPUs (DEVICE_TYPE_DISCRETE_GPU). On Android and iOS, the rendering device type is always considered to be DEVICE_TYPE_INTEGRATED_GPU.

DeviceType **DEVICE_TYPE_DISCRETE_GPU** = `2`

Rendering device is a dedicated GPU, which is typically *(but not always)* faster than integrated GPUs (DEVICE_TYPE_INTEGRATED_GPU).

DeviceType **DEVICE_TYPE_VIRTUAL_GPU** = `3`

Rendering device is an emulated GPU in a virtual environment. This is typically much slower than the host GPU, which means the expected performance level on a dedicated GPU will be roughly equivalent to DEVICE_TYPE_INTEGRATED_GPU. Virtual machine GPU passthrough (such as VFIO) will not report the device type as DEVICE_TYPE_VIRTUAL_GPU. Instead, the host GPU's device type will be reported as if the GPU was not emulated.

DeviceType **DEVICE_TYPE_CPU** = `4`

Rendering device is provided by software emulation (such as Lavapipe or SwiftShader [https://github.com/google/swiftshader]). This is the slowest kind of rendering device available; it's typically much slower than DEVICE_TYPE_INTEGRATED_GPU.

DeviceType **DEVICE_TYPE_MAX** = `5`

Represents the size of the DeviceType enum.

---

enum **DriverResource**: 🔗

DriverResource **DRIVER_RESOURCE_LOGICAL_DEVICE** = `0`

Specific device object based on a physical device.

- Vulkan: Vulkan device driver resource (`VkDevice`). (`rid` argument doesn't apply.)

DriverResource **DRIVER_RESOURCE_PHYSICAL_DEVICE** = `1`

Physical device the specific logical device is based on.

- Vulkan: `VkDevice`. (`rid` argument doesn't apply.)

DriverResource **DRIVER_RESOURCE_TOPMOST_OBJECT** = `2`

Top-most graphics API entry object.

- Vulkan: `VkInstance`. (`rid` argument doesn't apply.)

DriverResource **DRIVER_RESOURCE_COMMAND_QUEUE** = `3`

The main graphics-compute command queue.

- Vulkan: `VkQueue`. (`rid` argument doesn't apply.)

DriverResource **DRIVER_RESOURCE_QUEUE_FAMILY** = `4`

The specific family the main queue belongs to.

- Vulkan: the queue family index, an `uint32_t`. (`rid` argument doesn't apply.)

DriverResource **DRIVER_RESOURCE_TEXTURE** = `5`

- Vulkan: `VkImage`.

DriverResource **DRIVER_RESOURCE_TEXTURE_VIEW** = `6`

The view of an owned or shared texture.

- Vulkan: `VkImageView`.

DriverResource **DRIVER_RESOURCE_TEXTURE_DATA_FORMAT** = `7`

The native id of the data format of the texture.

- Vulkan: `VkFormat`.

DriverResource **DRIVER_RESOURCE_SAMPLER** = `8`

- Vulkan: `VkSampler`.

DriverResource **DRIVER_RESOURCE_UNIFORM_SET** = `9`

- Vulkan: `VkDescriptorSet`.

DriverResource **DRIVER_RESOURCE_BUFFER** = `10`

Buffer of any kind of (storage, vertex, etc.).

- Vulkan: `VkBuffer`.

DriverResource **DRIVER_RESOURCE_COMPUTE_PIPELINE** = `11`

- Vulkan: `VkPipeline`.

DriverResource **DRIVER_RESOURCE_RENDER_PIPELINE** = `12`

- Vulkan: `VkPipeline`.

DriverResource **DRIVER_RESOURCE_VULKAN_DEVICE** = `0`

Deprecated: Use DRIVER_RESOURCE_LOGICAL_DEVICE instead.

DriverResource **DRIVER_RESOURCE_VULKAN_PHYSICAL_DEVICE** = `1`

Deprecated: Use DRIVER_RESOURCE_PHYSICAL_DEVICE instead.

DriverResource **DRIVER_RESOURCE_VULKAN_INSTANCE** = `2`

Deprecated: Use DRIVER_RESOURCE_TOPMOST_OBJECT instead.

DriverResource **DRIVER_RESOURCE_VULKAN_QUEUE** = `3`

Deprecated: Use DRIVER_RESOURCE_COMMAND_QUEUE instead.

DriverResource **DRIVER_RESOURCE_VULKAN_QUEUE_FAMILY_INDEX** = `4`

Deprecated: Use DRIVER_RESOURCE_QUEUE_FAMILY instead.

DriverResource **DRIVER_RESOURCE_VULKAN_IMAGE** = `5`

Deprecated: Use DRIVER_RESOURCE_TEXTURE instead.

DriverResource **DRIVER_RESOURCE_VULKAN_IMAGE_VIEW** = `6`

Deprecated: Use DRIVER_RESOURCE_TEXTURE_VIEW instead.

DriverResource **DRIVER_RESOURCE_VULKAN_IMAGE_NATIVE_TEXTURE_FORMAT** = `7`

Deprecated: Use DRIVER_RESOURCE_TEXTURE_DATA_FORMAT instead.

DriverResource **DRIVER_RESOURCE_VULKAN_SAMPLER** = `8`

Deprecated: Use DRIVER_RESOURCE_SAMPLER instead.

DriverResource **DRIVER_RESOURCE_VULKAN_DESCRIPTOR_SET** = `9`

Deprecated: Use DRIVER_RESOURCE_UNIFORM_SET instead.

DriverResource **DRIVER_RESOURCE_VULKAN_BUFFER** = `10`

Deprecated: Use DRIVER_RESOURCE_BUFFER instead.

DriverResource **DRIVER_RESOURCE_VULKAN_COMPUTE_PIPELINE** = `11`

Deprecated: Use DRIVER_RESOURCE_COMPUTE_PIPELINE instead.

DriverResource **DRIVER_RESOURCE_VULKAN_RENDER_PIPELINE** = `12`

Deprecated: Use DRIVER_RESOURCE_RENDER_PIPELINE instead.

---

enum **DataFormat**: 🔗

DataFormat **DATA_FORMAT_R4G4_UNORM_PACK8** = `0`

4-bit-per-channel red/green channel data format, packed into 8 bits. Values are in the `[0.0, 1.0]` range.

Note: More information on all data formats can be found on the Identification of formats [https://registry.khronos.org/vulkan/specs/1.1/html/vkspec.html#_identification_of_formats] section of the Vulkan specification, as well as the VkFormat [https://registry.khronos.org/vulkan/specs/1.3-extensions/man/html/VkFormat.html] enum.

DataFormat **DATA_FORMAT_R4G4B4A4_UNORM_PACK16** = `1`

4-bit-per-channel red/green/blue/alpha channel data format, packed into 16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_B4G4R4A4_UNORM_PACK16** = `2`

4-bit-per-channel blue/green/red/alpha channel data format, packed into 16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R5G6B5_UNORM_PACK16** = `3`

Red/green/blue channel data format with 5 bits of red, 6 bits of green and 5 bits of blue, packed into 16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_B5G6R5_UNORM_PACK16** = `4`

Blue/green/red channel data format with 5 bits of blue, 6 bits of green and 5 bits of red, packed into 16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R5G5B5A1_UNORM_PACK16** = `5`

Red/green/blue/alpha channel data format with 5 bits of red, 6 bits of green, 5 bits of blue and 1 bit of alpha, packed into 16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_B5G5R5A1_UNORM_PACK16** = `6`

Blue/green/red/alpha channel data format with 5 bits of blue, 6 bits of green, 5 bits of red and 1 bit of alpha, packed into 16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_A1R5G5B5_UNORM_PACK16** = `7`

Alpha/red/green/blue channel data format with 1 bit of alpha, 5 bits of red, 6 bits of green and 5 bits of blue, packed into 16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8_UNORM** = `8`

8-bit-per-channel unsigned floating-point red channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8_SNORM** = `9`

8-bit-per-channel signed floating-point red channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8_USCALED** = `10`

8-bit-per-channel unsigned floating-point red channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 255.0]` range.

DataFormat **DATA_FORMAT_R8_SSCALED** = `11`

8-bit-per-channel signed floating-point red channel data format with scaled value (value is converted from integer to float). Values are in the `[-127.0, 127.0]` range.

DataFormat **DATA_FORMAT_R8_UINT** = `12`

8-bit-per-channel unsigned integer red channel data format. Values are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_R8_SINT** = `13`

8-bit-per-channel signed integer red channel data format. Values are in the `[-127, 127]` range.

DataFormat **DATA_FORMAT_R8_SRGB** = `14`

8-bit-per-channel unsigned floating-point red channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8G8_UNORM** = `15`

8-bit-per-channel unsigned floating-point red/green channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8G8_SNORM** = `16`

8-bit-per-channel signed floating-point red/green channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8G8_USCALED** = `17`

8-bit-per-channel unsigned floating-point red/green channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 255.0]` range.

DataFormat **DATA_FORMAT_R8G8_SSCALED** = `18`

8-bit-per-channel signed floating-point red/green channel data format with scaled value (value is converted from integer to float). Values are in the `[-127.0, 127.0]` range.

DataFormat **DATA_FORMAT_R8G8_UINT** = `19`

8-bit-per-channel unsigned integer red/green channel data format. Values are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_R8G8_SINT** = `20`

8-bit-per-channel signed integer red/green channel data format. Values are in the `[-127, 127]` range.

DataFormat **DATA_FORMAT_R8G8_SRGB** = `21`

8-bit-per-channel unsigned floating-point red/green channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8G8B8_UNORM** = `22`

8-bit-per-channel unsigned floating-point red/green/blue channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8G8B8_SNORM** = `23`

8-bit-per-channel signed floating-point red/green/blue channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8G8B8_USCALED** = `24`

8-bit-per-channel unsigned floating-point red/green/blue channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 255.0]` range.

DataFormat **DATA_FORMAT_R8G8B8_SSCALED** = `25`

8-bit-per-channel signed floating-point red/green/blue channel data format with scaled value (value is converted from integer to float). Values are in the `[-127.0, 127.0]` range.

DataFormat **DATA_FORMAT_R8G8B8_UINT** = `26`

8-bit-per-channel unsigned integer red/green/blue channel data format. Values are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_R8G8B8_SINT** = `27`

8-bit-per-channel signed integer red/green/blue channel data format. Values are in the `[-127, 127]` range.

DataFormat **DATA_FORMAT_R8G8B8_SRGB** = `28`

8-bit-per-channel unsigned floating-point red/green/blue/blue channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_B8G8R8_UNORM** = `29`

8-bit-per-channel unsigned floating-point blue/green/red channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_B8G8R8_SNORM** = `30`

8-bit-per-channel signed floating-point blue/green/red channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_B8G8R8_USCALED** = `31`

8-bit-per-channel unsigned floating-point blue/green/red channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 255.0]` range.

DataFormat **DATA_FORMAT_B8G8R8_SSCALED** = `32`

8-bit-per-channel signed floating-point blue/green/red channel data format with scaled value (value is converted from integer to float). Values are in the `[-127.0, 127.0]` range.

DataFormat **DATA_FORMAT_B8G8R8_UINT** = `33`

8-bit-per-channel unsigned integer blue/green/red channel data format. Values are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_B8G8R8_SINT** = `34`

8-bit-per-channel signed integer blue/green/red channel data format. Values are in the `[-127, 127]` range.

DataFormat **DATA_FORMAT_B8G8R8_SRGB** = `35`

8-bit-per-channel unsigned floating-point blue/green/red data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8G8B8A8_UNORM** = `36`

8-bit-per-channel unsigned floating-point red/green/blue/alpha channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8G8B8A8_SNORM** = `37`

8-bit-per-channel signed floating-point red/green/blue/alpha channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_R8G8B8A8_USCALED** = `38`

8-bit-per-channel unsigned floating-point red/green/blue/alpha channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 255.0]` range.

DataFormat **DATA_FORMAT_R8G8B8A8_SSCALED** = `39`

8-bit-per-channel signed floating-point red/green/blue/alpha channel data format with scaled value (value is converted from integer to float). Values are in the `[-127.0, 127.0]` range.

DataFormat **DATA_FORMAT_R8G8B8A8_UINT** = `40`

8-bit-per-channel unsigned integer red/green/blue/alpha channel data format. Values are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_R8G8B8A8_SINT** = `41`

8-bit-per-channel signed integer red/green/blue/alpha channel data format. Values are in the `[-127, 127]` range.

DataFormat **DATA_FORMAT_R8G8B8A8_SRGB** = `42`

8-bit-per-channel unsigned floating-point red/green/blue/alpha channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_B8G8R8A8_UNORM** = `43`

8-bit-per-channel unsigned floating-point blue/green/red/alpha channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_B8G8R8A8_SNORM** = `44`

8-bit-per-channel signed floating-point blue/green/red/alpha channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_B8G8R8A8_USCALED** = `45`

8-bit-per-channel unsigned floating-point blue/green/red/alpha channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 255.0]` range.

DataFormat **DATA_FORMAT_B8G8R8A8_SSCALED** = `46`

8-bit-per-channel signed floating-point blue/green/red/alpha channel data format with scaled value (value is converted from integer to float). Values are in the `[-127.0, 127.0]` range.

DataFormat **DATA_FORMAT_B8G8R8A8_UINT** = `47`

8-bit-per-channel unsigned integer blue/green/red/alpha channel data format. Values are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_B8G8R8A8_SINT** = `48`

8-bit-per-channel signed integer blue/green/red/alpha channel data format. Values are in the `[-127, 127]` range.

DataFormat **DATA_FORMAT_B8G8R8A8_SRGB** = `49`

8-bit-per-channel unsigned floating-point blue/green/red/alpha channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_A8B8G8R8_UNORM_PACK32** = `50`

8-bit-per-channel unsigned floating-point alpha/red/green/blue channel data format with normalized value, packed in 32 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_A8B8G8R8_SNORM_PACK32** = `51`

8-bit-per-channel signed floating-point alpha/red/green/blue channel data format with normalized value, packed in 32 bits. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_A8B8G8R8_USCALED_PACK32** = `52`

8-bit-per-channel unsigned floating-point alpha/red/green/blue channel data format with scaled value (value is converted from integer to float), packed in 32 bits. Values are in the `[0.0, 255.0]` range.

DataFormat **DATA_FORMAT_A8B8G8R8_SSCALED_PACK32** = `53`

8-bit-per-channel signed floating-point alpha/red/green/blue channel data format with scaled value (value is converted from integer to float), packed in 32 bits. Values are in the `[-127.0, 127.0]` range.

DataFormat **DATA_FORMAT_A8B8G8R8_UINT_PACK32** = `54`

8-bit-per-channel unsigned integer alpha/red/green/blue channel data format, packed in 32 bits. Values are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_A8B8G8R8_SINT_PACK32** = `55`

8-bit-per-channel signed integer alpha/red/green/blue channel data format, packed in 32 bits. Values are in the `[-127, 127]` range.

DataFormat **DATA_FORMAT_A8B8G8R8_SRGB_PACK32** = `56`

8-bit-per-channel unsigned floating-point alpha/red/green/blue channel data format with normalized value and non-linear sRGB encoding, packed in 32 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_A2R10G10B10_UNORM_PACK32** = `57`

Unsigned floating-point alpha/red/green/blue channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of red, 10 bits of green and 10 bits of blue. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_A2R10G10B10_SNORM_PACK32** = `58`

Signed floating-point alpha/red/green/blue channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of red, 10 bits of green and 10 bits of blue. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_A2R10G10B10_USCALED_PACK32** = `59`

Unsigned floating-point alpha/red/green/blue channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of red, 10 bits of green and 10 bits of blue. Values are in the `[0.0, 1023.0]` range for red/green/blue and `[0.0, 3.0]` for alpha.

DataFormat **DATA_FORMAT_A2R10G10B10_SSCALED_PACK32** = `60`

Signed floating-point alpha/red/green/blue channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of red, 10 bits of green and 10 bits of blue. Values are in the `[-511.0, 511.0]` range for red/green/blue and `[-1.0, 1.0]` for alpha.

DataFormat **DATA_FORMAT_A2R10G10B10_UINT_PACK32** = `61`

Unsigned integer alpha/red/green/blue channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of red, 10 bits of green and 10 bits of blue. Values are in the `[0, 1023]` range for red/green/blue and `[0, 3]` for alpha.

DataFormat **DATA_FORMAT_A2R10G10B10_SINT_PACK32** = `62`

Signed integer alpha/red/green/blue channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of red, 10 bits of green and 10 bits of blue. Values are in the `[-511, 511]` range for red/green/blue and `[-1, 1]` for alpha.

DataFormat **DATA_FORMAT_A2B10G10R10_UNORM_PACK32** = `63`

Unsigned floating-point alpha/blue/green/red channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of blue, 10 bits of green and 10 bits of red. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_A2B10G10R10_SNORM_PACK32** = `64`

Signed floating-point alpha/blue/green/red channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of blue, 10 bits of green and 10 bits of red. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_A2B10G10R10_USCALED_PACK32** = `65`

Unsigned floating-point alpha/blue/green/red channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of blue, 10 bits of green and 10 bits of red. Values are in the `[0.0, 1023.0]` range for blue/green/red and `[0.0, 3.0]` for alpha.

DataFormat **DATA_FORMAT_A2B10G10R10_SSCALED_PACK32** = `66`

Signed floating-point alpha/blue/green/red channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of blue, 10 bits of green and 10 bits of red. Values are in the `[-511.0, 511.0]` range for blue/green/red and `[-1.0, 1.0]` for alpha.

DataFormat **DATA_FORMAT_A2B10G10R10_UINT_PACK32** = `67`

Unsigned integer alpha/blue/green/red channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of blue, 10 bits of green and 10 bits of red. Values are in the `[0, 1023]` range for blue/green/red and `[0, 3]` for alpha.

DataFormat **DATA_FORMAT_A2B10G10R10_SINT_PACK32** = `68`

Signed integer alpha/blue/green/red channel data format with normalized value, packed in 32 bits. Format contains 2 bits of alpha, 10 bits of blue, 10 bits of green and 10 bits of red. Values are in the `[-511, 511]` range for blue/green/red and `[-1, 1]` for alpha.

DataFormat **DATA_FORMAT_R16_UNORM** = `69`

16-bit-per-channel unsigned floating-point red channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R16_SNORM** = `70`

16-bit-per-channel signed floating-point red channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_R16_USCALED** = `71`

16-bit-per-channel unsigned floating-point red channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 65535.0]` range.

DataFormat **DATA_FORMAT_R16_SSCALED** = `72`

16-bit-per-channel signed floating-point red channel data format with scaled value (value is converted from integer to float). Values are in the `[-32767.0, 32767.0]` range.

DataFormat **DATA_FORMAT_R16_UINT** = `73`

16-bit-per-channel unsigned integer red channel data format. Values are in the `[0.0, 65535]` range.

DataFormat **DATA_FORMAT_R16_SINT** = `74`

16-bit-per-channel signed integer red channel data format. Values are in the `[-32767, 32767]` range.

DataFormat **DATA_FORMAT_R16_SFLOAT** = `75`

16-bit-per-channel signed floating-point red channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R16G16_UNORM** = `76`

16-bit-per-channel unsigned floating-point red/green channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R16G16_SNORM** = `77`

16-bit-per-channel signed floating-point red/green channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_R16G16_USCALED** = `78`

16-bit-per-channel unsigned floating-point red/green channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 65535.0]` range.

DataFormat **DATA_FORMAT_R16G16_SSCALED** = `79`

16-bit-per-channel signed floating-point red/green channel data format with scaled value (value is converted from integer to float). Values are in the `[-32767.0, 32767.0]` range.

DataFormat **DATA_FORMAT_R16G16_UINT** = `80`

16-bit-per-channel unsigned integer red/green channel data format. Values are in the `[0.0, 65535]` range.

DataFormat **DATA_FORMAT_R16G16_SINT** = `81`

16-bit-per-channel signed integer red/green channel data format. Values are in the `[-32767, 32767]` range.

DataFormat **DATA_FORMAT_R16G16_SFLOAT** = `82`

16-bit-per-channel signed floating-point red/green channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R16G16B16_UNORM** = `83`

16-bit-per-channel unsigned floating-point red/green/blue channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R16G16B16_SNORM** = `84`

16-bit-per-channel signed floating-point red/green/blue channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_R16G16B16_USCALED** = `85`

16-bit-per-channel unsigned floating-point red/green/blue channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 65535.0]` range.

DataFormat **DATA_FORMAT_R16G16B16_SSCALED** = `86`

16-bit-per-channel signed floating-point red/green/blue channel data format with scaled value (value is converted from integer to float). Values are in the `[-32767.0, 32767.0]` range.

DataFormat **DATA_FORMAT_R16G16B16_UINT** = `87`

16-bit-per-channel unsigned integer red/green/blue channel data format. Values are in the `[0.0, 65535]` range.

DataFormat **DATA_FORMAT_R16G16B16_SINT** = `88`

16-bit-per-channel signed integer red/green/blue channel data format. Values are in the `[-32767, 32767]` range.

DataFormat **DATA_FORMAT_R16G16B16_SFLOAT** = `89`

16-bit-per-channel signed floating-point red/green/blue channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R16G16B16A16_UNORM** = `90`

16-bit-per-channel unsigned floating-point red/green/blue/alpha channel data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R16G16B16A16_SNORM** = `91`

16-bit-per-channel signed floating-point red/green/blue/alpha channel data format with normalized value. Values are in the `[-1.0, 1.0]` range.

DataFormat **DATA_FORMAT_R16G16B16A16_USCALED** = `92`

16-bit-per-channel unsigned floating-point red/green/blue/alpha channel data format with scaled value (value is converted from integer to float). Values are in the `[0.0, 65535.0]` range.

DataFormat **DATA_FORMAT_R16G16B16A16_SSCALED** = `93`

16-bit-per-channel signed floating-point red/green/blue/alpha channel data format with scaled value (value is converted from integer to float). Values are in the `[-32767.0, 32767.0]` range.

DataFormat **DATA_FORMAT_R16G16B16A16_UINT** = `94`

16-bit-per-channel unsigned integer red/green/blue/alpha channel data format. Values are in the `[0.0, 65535]` range.

DataFormat **DATA_FORMAT_R16G16B16A16_SINT** = `95`

16-bit-per-channel signed integer red/green/blue/alpha channel data format. Values are in the `[-32767, 32767]` range.

DataFormat **DATA_FORMAT_R16G16B16A16_SFLOAT** = `96`

16-bit-per-channel signed floating-point red/green/blue/alpha channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R32_UINT** = `97`

32-bit-per-channel unsigned integer red channel data format. Values are in the `[0, 2^32 - 1]` range.

DataFormat **DATA_FORMAT_R32_SINT** = `98`

32-bit-per-channel signed integer red channel data format. Values are in the `[2^31 + 1, 2^31 - 1]` range.

DataFormat **DATA_FORMAT_R32_SFLOAT** = `99`

32-bit-per-channel signed floating-point red channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R32G32_UINT** = `100`

32-bit-per-channel unsigned integer red/green channel data format. Values are in the `[0, 2^32 - 1]` range.

DataFormat **DATA_FORMAT_R32G32_SINT** = `101`

32-bit-per-channel signed integer red/green channel data format. Values are in the `[2^31 + 1, 2^31 - 1]` range.

DataFormat **DATA_FORMAT_R32G32_SFLOAT** = `102`

32-bit-per-channel signed floating-point red/green channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R32G32B32_UINT** = `103`

32-bit-per-channel unsigned integer red/green/blue channel data format. Values are in the `[0, 2^32 - 1]` range.

DataFormat **DATA_FORMAT_R32G32B32_SINT** = `104`

32-bit-per-channel signed integer red/green/blue channel data format. Values are in the `[2^31 + 1, 2^31 - 1]` range.

DataFormat **DATA_FORMAT_R32G32B32_SFLOAT** = `105`

32-bit-per-channel signed floating-point red/green/blue channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R32G32B32A32_UINT** = `106`

32-bit-per-channel unsigned integer red/green/blue/alpha channel data format. Values are in the `[0, 2^32 - 1]` range.

DataFormat **DATA_FORMAT_R32G32B32A32_SINT** = `107`

32-bit-per-channel signed integer red/green/blue/alpha channel data format. Values are in the `[2^31 + 1, 2^31 - 1]` range.

DataFormat **DATA_FORMAT_R32G32B32A32_SFLOAT** = `108`

32-bit-per-channel signed floating-point red/green/blue/alpha channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R64_UINT** = `109`

64-bit-per-channel unsigned integer red channel data format. Values are in the `[0, 2^64 - 1]` range.

DataFormat **DATA_FORMAT_R64_SINT** = `110`

64-bit-per-channel signed integer red channel data format. Values are in the `[2^63 + 1, 2^63 - 1]` range.

DataFormat **DATA_FORMAT_R64_SFLOAT** = `111`

64-bit-per-channel signed floating-point red channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R64G64_UINT** = `112`

64-bit-per-channel unsigned integer red/green channel data format. Values are in the `[0, 2^64 - 1]` range.

DataFormat **DATA_FORMAT_R64G64_SINT** = `113`

64-bit-per-channel signed integer red/green channel data format. Values are in the `[2^63 + 1, 2^63 - 1]` range.

DataFormat **DATA_FORMAT_R64G64_SFLOAT** = `114`

64-bit-per-channel signed floating-point red/green channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R64G64B64_UINT** = `115`

64-bit-per-channel unsigned integer red/green/blue channel data format. Values are in the `[0, 2^64 - 1]` range.

DataFormat **DATA_FORMAT_R64G64B64_SINT** = `116`

64-bit-per-channel signed integer red/green/blue channel data format. Values are in the `[2^63 + 1, 2^63 - 1]` range.

DataFormat **DATA_FORMAT_R64G64B64_SFLOAT** = `117`

64-bit-per-channel signed floating-point red/green/blue channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_R64G64B64A64_UINT** = `118`

64-bit-per-channel unsigned integer red/green/blue/alpha channel data format. Values are in the `[0, 2^64 - 1]` range.

DataFormat **DATA_FORMAT_R64G64B64A64_SINT** = `119`

64-bit-per-channel signed integer red/green/blue/alpha channel data format. Values are in the `[2^63 + 1, 2^63 - 1]` range.

DataFormat **DATA_FORMAT_R64G64B64A64_SFLOAT** = `120`

64-bit-per-channel signed floating-point red/green/blue/alpha channel data format with the value stored as-is.

DataFormat **DATA_FORMAT_B10G11R11_UFLOAT_PACK32** = `121`

Unsigned floating-point blue/green/red data format with the value stored as-is, packed in 32 bits. The format's precision is 10 bits of blue channel, 11 bits of green channel and 11 bits of red channel.

DataFormat **DATA_FORMAT_E5B9G9R9_UFLOAT_PACK32** = `122`

Unsigned floating-point exposure/blue/green/red data format with the value stored as-is, packed in 32 bits. The format's precision is 5 bits of exposure, 9 bits of blue channel, 9 bits of green channel and 9 bits of red channel.

DataFormat **DATA_FORMAT_D16_UNORM** = `123`

16-bit unsigned floating-point depth data format with normalized value. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_X8_D24_UNORM_PACK32** = `124`

24-bit unsigned floating-point depth data format with normalized value, plus 8 unused bits, packed in 32 bits. Values for depth are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_D32_SFLOAT** = `125`

32-bit signed floating-point depth data format with the value stored as-is.

DataFormat **DATA_FORMAT_S8_UINT** = `126`

8-bit unsigned integer stencil data format.

DataFormat **DATA_FORMAT_D16_UNORM_S8_UINT** = `127`

16-bit unsigned floating-point depth data format with normalized value, plus 8 bits of stencil in unsigned integer format. Values for depth are in the `[0.0, 1.0]` range. Values for stencil are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_D24_UNORM_S8_UINT** = `128`

24-bit unsigned floating-point depth data format with normalized value, plus 8 bits of stencil in unsigned integer format. Values for depth are in the `[0.0, 1.0]` range. Values for stencil are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_D32_SFLOAT_S8_UINT** = `129`

32-bit signed floating-point depth data format with the value stored as-is, plus 8 bits of stencil in unsigned integer format. Values for stencil are in the `[0, 255]` range.

DataFormat **DATA_FORMAT_BC1_RGB_UNORM_BLOCK** = `130`

VRAM-compressed unsigned red/green/blue channel data format with normalized value. Values are in the `[0.0, 1.0]` range. The format's precision is 5 bits of red channel, 6 bits of green channel and 5 bits of blue channel. Using BC1 texture compression (also known as S3TC DXT1).

DataFormat **DATA_FORMAT_BC1_RGB_SRGB_BLOCK** = `131`

VRAM-compressed unsigned red/green/blue channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range. The format's precision is 5 bits of red channel, 6 bits of green channel and 5 bits of blue channel. Using BC1 texture compression (also known as S3TC DXT1).

DataFormat **DATA_FORMAT_BC1_RGBA_UNORM_BLOCK** = `132`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value. Values are in the `[0.0, 1.0]` range. The format's precision is 5 bits of red channel, 6 bits of green channel, 5 bits of blue channel and 1 bit of alpha channel. Using BC1 texture compression (also known as S3TC DXT1).

DataFormat **DATA_FORMAT_BC1_RGBA_SRGB_BLOCK** = `133`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range. The format's precision is 5 bits of red channel, 6 bits of green channel, 5 bits of blue channel and 1 bit of alpha channel. Using BC1 texture compression (also known as S3TC DXT1).

DataFormat **DATA_FORMAT_BC2_UNORM_BLOCK** = `134`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value. Values are in the `[0.0, 1.0]` range. The format's precision is 5 bits of red channel, 6 bits of green channel, 5 bits of blue channel and 4 bits of alpha channel. Using BC2 texture compression (also known as S3TC DXT3).

DataFormat **DATA_FORMAT_BC2_SRGB_BLOCK** = `135`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range. The format's precision is 5 bits of red channel, 6 bits of green channel, 5 bits of blue channel and 4 bits of alpha channel. Using BC2 texture compression (also known as S3TC DXT3).

DataFormat **DATA_FORMAT_BC3_UNORM_BLOCK** = `136`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value. Values are in the `[0.0, 1.0]` range. The format's precision is 5 bits of red channel, 6 bits of green channel, 5 bits of blue channel and 8 bits of alpha channel. Using BC3 texture compression (also known as S3TC DXT5).

DataFormat **DATA_FORMAT_BC3_SRGB_BLOCK** = `137`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range. The format's precision is 5 bits of red channel, 6 bits of green channel, 5 bits of blue channel and 8 bits of alpha channel. Using BC3 texture compression (also known as S3TC DXT5).

DataFormat **DATA_FORMAT_BC4_UNORM_BLOCK** = `138`

VRAM-compressed unsigned red channel data format with normalized value. Values are in the `[0.0, 1.0]` range. The format's precision is 8 bits of red channel. Using BC4 texture compression.

DataFormat **DATA_FORMAT_BC4_SNORM_BLOCK** = `139`

VRAM-compressed signed red channel data format with normalized value. Values are in the `[-1.0, 1.0]` range. The format's precision is 8 bits of red channel. Using BC4 texture compression.

DataFormat **DATA_FORMAT_BC5_UNORM_BLOCK** = `140`

VRAM-compressed unsigned red/green channel data format with normalized value. Values are in the `[0.0, 1.0]` range. The format's precision is 8 bits of red channel and 8 bits of green channel. Using BC5 texture compression (also known as S3TC RGTC).

DataFormat **DATA_FORMAT_BC5_SNORM_BLOCK** = `141`

VRAM-compressed signed red/green channel data format with normalized value. Values are in the `[-1.0, 1.0]` range. The format's precision is 8 bits of red channel and 8 bits of green channel. Using BC5 texture compression (also known as S3TC RGTC).

DataFormat **DATA_FORMAT_BC6H_UFLOAT_BLOCK** = `142`

VRAM-compressed unsigned red/green/blue channel data format with the floating-point value stored as-is. The format's precision is between 10 and 13 bits for the red/green/blue channels. Using BC6H texture compression (also known as BPTC HDR).

DataFormat **DATA_FORMAT_BC6H_SFLOAT_BLOCK** = `143`

VRAM-compressed signed red/green/blue channel data format with the floating-point value stored as-is. The format's precision is between 10 and 13 bits for the red/green/blue channels. Using BC6H texture compression (also known as BPTC HDR).

DataFormat **DATA_FORMAT_BC7_UNORM_BLOCK** = `144`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value. Values are in the `[0.0, 1.0]` range. The format's precision is between 4 and 7 bits for the red/green/blue channels and between 0 and 8 bits for the alpha channel. Also known as BPTC LDR.

DataFormat **DATA_FORMAT_BC7_SRGB_BLOCK** = `145`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range. The format's precision is between 4 and 7 bits for the red/green/blue channels and between 0 and 8 bits for the alpha channel. Also known as BPTC LDR.

DataFormat **DATA_FORMAT_ETC2_R8G8B8_UNORM_BLOCK** = `146`

VRAM-compressed unsigned red/green/blue channel data format with normalized value. Values are in the `[0.0, 1.0]` range. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_ETC2_R8G8B8_SRGB_BLOCK** = `147`

VRAM-compressed unsigned red/green/blue channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_ETC2_R8G8B8A1_UNORM_BLOCK** = `148`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value. Values are in the `[0.0, 1.0]` range. Red/green/blue use 8 bit of precision each, with alpha using 1 bit of precision. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_ETC2_R8G8B8A1_SRGB_BLOCK** = `149`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value and non-linear sRGB encoding. Values are in the `[0.0, 1.0]` range. Red/green/blue use 8 bit of precision each, with alpha using 1 bit of precision. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_ETC2_R8G8B8A8_UNORM_BLOCK** = `150`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value. Values are in the `[0.0, 1.0]` range. Red/green/blue use 8 bits of precision each, with alpha using 8 bits of precision. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_ETC2_R8G8B8A8_SRGB_BLOCK** = `151`

VRAM-compressed unsigned red/green/blue/alpha channel data format with normalized value and non-linear sRGB encoding. Values are in the

`[0.0, 1.0]` range. Red/green/blue use 8 bits of precision each, with alpha using 8 bits of precision. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_EAC_R11_UNORM_BLOCK** = `152`

11-bit VRAM-compressed unsigned red channel data format with normalized value. Values are in the `[0.0, 1.0]` range. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_EAC_R11_SNORM_BLOCK** = `153`

11-bit VRAM-compressed signed red channel data format with normalized value. Values are in the `[-1.0, 1.0]` range. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_EAC_R11G11_UNORM_BLOCK** = `154`

11-bit VRAM-compressed unsigned red/green channel data format with normalized value. Values are in the `[0.0, 1.0]` range. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_EAC_R11G11_SNORM_BLOCK** = `155`

11-bit VRAM-compressed signed red/green channel data format with normalized value. Values are in the `[-1.0, 1.0]` range. Using ETC2 texture compression.

DataFormat **DATA_FORMAT_ASTC_4x4_UNORM_BLOCK** = `156`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 4×4 blocks (highest quality). Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_4x4_SRGB_BLOCK** = `157`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 4×4 blocks (highest quality). Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_5x4_UNORM_BLOCK** = `158`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 5×4 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_5x4_SRGB_BLOCK** = `159`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 5×4 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_5x5_UNORM_BLOCK** = `160`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 5×5 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_5x5_SRGB_BLOCK** = `161`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 5×5 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_6x5_UNORM_BLOCK** = `162`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 6×5 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_6x5_SRGB_BLOCK** = `163`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 6×5 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_6x6_UNORM_BLOCK** = `164`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 6×6 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_6x6_SRGB_BLOCK** = `165`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 6×6 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_8x5_UNORM_BLOCK** = `166`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 8×5 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_8x5_SRGB_BLOCK** = `167`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 8×5 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_8x6_UNORM_BLOCK** = `168`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 8×6 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_8x6_SRGB_BLOCK** = `169`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 8×6 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_8x8_UNORM_BLOCK** = `170`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 8×8 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_8x8_SRGB_BLOCK** = `171`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 8×8 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_10x5_UNORM_BLOCK** = `172`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 10×5 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_10x5_SRGB_BLOCK** = `173`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 10×5 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_10x6_UNORM_BLOCK** = `174`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 10×6 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_10x6_SRGB_BLOCK** = `175`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 10×6 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_10x8_UNORM_BLOCK** = `176`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 10×8 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_10x8_SRGB_BLOCK** = `177`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 10×8 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_10x10_UNORM_BLOCK** = `178`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 10×10 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_10x10_SRGB_BLOCK** = `179`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 10×10 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_12x10_UNORM_BLOCK** = `180`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 12×10 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_12x10_SRGB_BLOCK** = `181`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 12×10 blocks. Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_12x12_UNORM_BLOCK** = `182`

VRAM-compressed unsigned floating-point data format with normalized value, packed in 12 blocks (lowest quality). Values are in the `[0.0,` `1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_ASTC_12x12_SRGB_BLOCK** = `183`

VRAM-compressed unsigned floating-point data format with normalized value and non-linear sRGB encoding, packed in 12 blocks (lowest quality). Values are in the `[0.0, 1.0]` range. Using ASTC compression.

DataFormat **DATA_FORMAT_G8B8G8R8_422_UNORM** = `184`

8-bit-per-channel unsigned floating-point green/blue/red channel data format with normalized value. Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_B8G8R8G8_422_UNORM** = `185`

8-bit-per-channel unsigned floating-point blue/green/red channel data format with normalized value. Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G8_B8_R8_3PLANE_420_UNORM** = `186`

8-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, stored across 3 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal and vertical resolution (i.e. 2×2 adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G8_B8R8_2PLANE_420_UNORM** = `187`

8-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, stored across 2 separate planes (green +

blue/red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal and vertical resolution (i.e. 2×2 adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G8_B8_R8_3PLANE_422_UNORM** = `188`

8-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, stored across 2 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G8_B8R8_2PLANE_422_UNORM** = `189`

8-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, stored across 2 separate planes (green + blue/red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G8_B8_R8_3PLANE_444_UNORM** = `190`

8-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, stored across 3 separate planes. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R10X6_UNORM_PACK16** = `191`

10-bit-per-channel unsigned floating-point red channel data with normalized value, plus 6 unused bits, packed in 16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R10X6G10X6_UNORM_2PACK16** = `192`

10-bit-per-channel unsigned floating-point red/green channel data with normalized value, plus 6 unused bits after each channel, packed in 2×16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R10X6G10X6B10X6A10X6_UNORM_4PACK16** = `193`

10-bit-per-channel unsigned floating-point red/green/blue/alpha channel data with normalized value, plus 6 unused bits after each channel, packed in 4×16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_G10X6B10X6G10X6R10X6_422_UNORM_4PACK16** = `194`

10-bit-per-channel unsigned floating-point green/blue/green/red channel data with normalized value, plus 6 unused bits after each channel, packed in 4×16 bits. Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel). The green channel is listed twice, but contains different values to allow it to be represented at full resolution.

DataFormat **DATA_FORMAT_B10X6G10X6R10X6G10X6_422_UNORM_4PACK16** = `195`

10-bit-per-channel unsigned floating-point blue/green/red/green channel data with normalized value, plus 6 unused bits after each channel, packed in 4×16 bits. Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel). The green channel is listed twice, but contains different values to allow it to be represented at full resolution.

DataFormat **DATA_FORMAT_G10X6_B10X6_R10X6_3PLANE_420_UNORM**

_3PACK16 = `196`

10-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 2 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal and vertical resolution (i.e. 2×2 adjacent pixels will share the same value for the blue/red channel).

DataFormat
DATA_FORMAT_G10X6_B10X6R10X6_2PLANE_420_UNORM_3PACK16 = `197`

10-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 2 separate planes (green + blue/red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal and vertical resolution (i.e. 2×2 adjacent pixels will share the same value for the blue/red channel).

DataFormat
DATA_FORMAT_G10X6_B10X6_R10X6_3PLANE_422_UNORM_3PACK16 = `198`

10-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 3 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat
DATA_FORMAT_G10X6_B10X6R10X6_2PLANE_422_UNORM_3PACK16 = `199`

10-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 3 separate planes (green + blue/red).

Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G10X6_B10X6_R10X6_3PLANE_444_UNORM_3PACK16** = `200`

10-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 3 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R12X4_UNORM_PACK16** = `201`

12-bit-per-channel unsigned floating-point red channel data with normalized value, plus 6 unused bits, packed in 16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R12X4G12X4_UNORM_2PACK16** = `202`

12-bit-per-channel unsigned floating-point red/green channel data with normalized value, plus 6 unused bits after each channel, packed in 2×16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_R12X4G12X4B12X4A12X4_UNORM_4PACK16** = `203`

12-bit-per-channel unsigned floating-point red/green/blue/alpha channel data with normalized value, plus 6 unused bits after each channel, packed in 4×16 bits. Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_G12X4B12X4G12X4R12X4_422_UNORM_4PACK16** = `204`

12-bit-per-channel unsigned floating-point green/blue/green/red channel data with normalized value, plus 6 unused bits after each channel, packed in 4×16 bits. Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel). The green channel is listed twice, but contains different values to allow it to be represented at full resolution.

DataFormat **DATA_FORMAT_B12X4G12X4R12X4G12X4_422_UNORM_4PACK16** = `205`

12-bit-per-channel unsigned floating-point blue/green/red/green channel data with normalized value, plus 6 unused bits after each channel, packed in 4×16 bits. Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel). The green channel is listed twice, but contains different values to allow it to be represented at full resolution.

DataFormat **DATA_FORMAT_G12X4_B12X4_R12X4_3PLANE_420_UNORM_3PACK16** = `206`

12-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 2 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal and vertical resolution (i.e. 2×2 adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G12X4_B12X4R12X4_2PLANE_420_UNORM_3PACK16** = `207`

12-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 2 separate planes (green + blue/red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal and vertical resolution (i.e. 2×2 adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G12X4_B12X4_R12X4_3PLANE_422_UNORM_3PACK16** = `208`

12-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 3 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G12X4_B12X4R12X4_2PLANE_422_UNORM_3PACK16** = `209`

12-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 3 separate planes (green + blue/red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G12X4_B12X4_R12X4_3PLANE_444_UNORM_3PACK16** = `210`

12-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Packed in 3×16 bits and stored across 3 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_G16B16G16R16_422_UNORM** = `211`

16-bit-per-channel unsigned floating-point green/blue/red channel data format with normalized value. Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_B16G16R16G16_422_UNORM** = `212`

16-bit-per-channel unsigned floating-point blue/green/red channel data format with normalized value. Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G16_B16_R16_3PLANE_420_UNORM** = `213`

16-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Stored across 2 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal and vertical resolution (i.e. 2×2 adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G16_B16R16_2PLANE_420_UNORM** = `214`

16-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Stored across 2 separate planes (green + blue/red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal and vertical resolution (i.e. 2×2 adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G16_B16_R16_3PLANE_422_UNORM** = `215`

16-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Stored across 3 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G16_B16R16_2PLANE_422_UNORM** = `216`

16-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Stored across 3 separate planes (green + blue/red). Values are in the `[0.0, 1.0]` range. Blue and red channel data is stored at halved horizontal resolution (i.e. 2 horizontally adjacent pixels will share the same value for the blue/red channel).

DataFormat **DATA_FORMAT_G16_B16_R16_3PLANE_444_UNORM** = `217`

16-bit-per-channel unsigned floating-point green/blue/red channel data with normalized value, plus 6 unused bits after each channel. Stored across 3 separate planes (green + blue + red). Values are in the `[0.0, 1.0]` range.

DataFormat **DATA_FORMAT_MAX** = `218`

Represents the size of the DataFormat enum.

---

flags **BarrierMask**: 🔗

BarrierMask **BARRIER_MASK_VERTEX** = `1`

Vertex shader barrier mask.

BarrierMask **BARRIER_MASK_FRAGMENT** = `8`

Fragment shader barrier mask.

BarrierMask **BARRIER_MASK_COMPUTE** = `2`

Compute barrier mask.

BarrierMask **BARRIER_MASK_TRANSFER** = `4`

Transfer barrier mask.

BarrierMask **BARRIER_MASK_RASTER** = `9`

Raster barrier mask (vertex and fragment). Equivalent to `BARRIER_MASK_VERTEX | BARRIER_MASK_FRAGMENT`.

BarrierMask **BARRIER_MASK_ALL_BARRIERS** = `32767`

Barrier mask for all types (vertex, fragment, compute, transfer).

BarrierMask **BARRIER_MASK_NO_BARRIER** = `32768`

No barrier for any type.

---

enum **TextureType**: 🔗

TextureType **TEXTURE_TYPE_1D** = `0`

1-dimensional texture.

TextureType **TEXTURE_TYPE_2D** = `1`

2-dimensional texture.

TextureType **TEXTURE_TYPE_3D** = `2`

3-dimensional texture.

TextureType **TEXTURE_TYPE_CUBE** = `3`

Cubemap texture.

TextureType **TEXTURE_TYPE_1D_ARRAY** = `4`

Array of 1-dimensional textures.

TextureType **TEXTURE_TYPE_2D_ARRAY** = `5`

Array of 2-dimensional textures.

TextureType **TEXTURE_TYPE_CUBE_ARRAY** = `6`

Array of Cubemap textures.

TextureType **TEXTURE_TYPE_MAX** = `7`

Represents the size of the TextureType enum.

enum **TextureSamples**: 🔗

TextureSamples **TEXTURE_SAMPLES_1** = `0`

Perform 1 texture sample (this is the fastest but lowest-quality for antialiasing).

TextureSamples **TEXTURE_SAMPLES_2** = `1`

Perform 2 texture samples.

TextureSamples **TEXTURE_SAMPLES_4** = `2`

Perform 4 texture samples.

TextureSamples **TEXTURE_SAMPLES_8** = `3`

Perform 8 texture samples. Not supported on mobile GPUs (including Apple Silicon).

TextureSamples **TEXTURE_SAMPLES_16** = `4`

Perform 16 texture samples. Not supported on mobile GPUs and many desktop GPUs.

TextureSamples **TEXTURE_SAMPLES_32** = `5`

Perform 32 texture samples. Not supported on most GPUs.

TextureSamples **TEXTURE_SAMPLES_64** = `6`

Perform 64 texture samples (this is the slowest but highest-quality for antialiasing). Not supported on most GPUs.

TextureSamples **TEXTURE_SAMPLES_MAX** = `7`

Represents the size of the TextureSamples enum.

---

flags **TextureUsageBits**: 🔗

TextureUsageBits **TEXTURE_USAGE_SAMPLING_BIT** = `1`

Texture can be sampled.

TextureUsageBits **TEXTURE_USAGE_COLOR_ATTACHMENT_BIT** = `2`

Texture can be used as a color attachment in a framebuffer.

TextureUsageBits **TEXTURE_USAGE_DEPTH_STENCIL_ATTACHMENT_BIT** = `4`

Texture can be used as a depth/stencil attachment in a framebuffer.

TextureUsageBits **TEXTURE_USAGE_STORAGE_BIT** = `8`

Texture can be used as a storage image [https://registry.khronos.org/vulkan/specs/1.3-extensions/html/vkspec.html#descriptorsets-storageimage].

TextureUsageBits **TEXTURE_USAGE_STORAGE_ATOMIC_BIT** = `16`

Texture can be used as a storage image [https://registry.khronos.org/vulkan/specs/1.3-extensions/html/vkspec.html#descriptorsets-storageimage] with support for atomic operations.

TextureUsageBits **TEXTURE_USAGE_CPU_READ_BIT** = `32`

Texture can be read back on the CPU using texture_get_data faster than without this bit, since it is always kept in the system memory.

TextureUsageBits **TEXTURE_USAGE_CAN_UPDATE_BIT** = `64`

Texture can be updated using texture_update.

TextureUsageBits **TEXTURE_USAGE_CAN_COPY_FROM_BIT** = `128`

Texture can be a source for texture_copy.

TextureUsageBits **TEXTURE_USAGE_CAN_COPY_TO_BIT** = `256`

Texture can be a destination for texture_copy.

TextureUsageBits **TEXTURE_USAGE_INPUT_ATTACHMENT_BIT** = `512`

Texture can be used as a input attachment [https://registry.khronos.org/vulkan/specs/1.3-extensions/html/vkspec.html#descriptorsets-inputattachment] in a framebuffer.

---

enum **TextureSwizzle**: 🔗

TextureSwizzle **TEXTURE_SWIZZLE_IDENTITY** = `0`

Return the sampled value as-is.

TextureSwizzle **TEXTURE_SWIZZLE_ZERO** = `1`

Always return `0.0` when sampling.

TextureSwizzle **TEXTURE_SWIZZLE_ONE** = `2`

Always return `1.0` when sampling.

TextureSwizzle **TEXTURE_SWIZZLE_R** = `3`

Sample the red color channel.

TextureSwizzle **TEXTURE_SWIZZLE_G** = `4`

Sample the green color channel.

TextureSwizzle **TEXTURE_SWIZZLE_B** = `5`

Sample the blue color channel.

TextureSwizzle **TEXTURE_SWIZZLE_A** = `6`

Sample the alpha channel.

TextureSwizzle **TEXTURE_SWIZZLE_MAX** = `7`

Represents the size of the TextureSwizzle enum.

---

enum **TextureSliceType**: 🔗

TextureSliceType **TEXTURE_SLICE_2D** = `0`

2-dimensional texture slice.

TextureSliceType **TEXTURE_SLICE_CUBEMAP** = `1`

Cubemap texture slice.

TextureSliceType **TEXTURE_SLICE_3D** = `2`

3-dimensional texture slice.

---

enum **SamplerFilter**: 🔗

SamplerFilter **SAMPLER_FILTER_NEAREST** = `0`

Nearest-neighbor sampler filtering. Sampling at higher resolutions than the source will result in a pixelated look.

SamplerFilter **SAMPLER_FILTER_LINEAR** = `1`

Bilinear sampler filtering. Sampling at higher resolutions than the source will result in a blurry look.

---

enum **SamplerRepeatMode**: 🔗

SamplerRepeatMode **SAMPLER_REPEAT_MODE_REPEAT** = `0`

Sample with repeating enabled.

SamplerRepeatMode **SAMPLER_REPEAT_MODE_MIRRORED_REPEAT** = `1`

Sample with mirrored repeating enabled. When sampling outside the `[0.0, 1.0]` range, return a mirrored version of the sampler. This mirrored version is mirrored again if sampling further away, with the pattern repeating indefinitely.

SamplerRepeatMode **SAMPLER_REPEAT_MODE_CLAMP_TO_EDGE** = `2`

Sample with repeating disabled. When sampling outside the `[0.0, 1.0]` range, return the color of the last pixel on the edge.

SamplerRepeatMode **SAMPLER_REPEAT_MODE_CLAMP_TO_BORDER** = `3`

Sample with repeating disabled. When sampling outside the `[0.0, 1.0]` range, return the specified RDSamplerState.border_color.

SamplerRepeatMode **SAMPLER_REPEAT_MODE_MIRROR_CLAMP_TO_EDGE** = `4`

Sample with mirrored repeating enabled, but only once. When sampling in the `[-1.0, 0.0]` range, return a mirrored version of the sampler. When sampling outside the `[-1.0, 1.0]` range, return the color of the last pixel on the edge.

SamplerRepeatMode **SAMPLER_REPEAT_MODE_MAX** = `5`

Represents the size of the SamplerRepeatMode enum.

---

enum **SamplerBorderColor**: 🔗

SamplerBorderColor **SAMPLER_BORDER_COLOR_FLOAT_TRANSPARENT_BLACK** = `0`

Return a floating-point transparent black color when sampling outside the `[0.0, 1.0]` range. Only effective if the sampler repeat mode is SAMPLER_REPEAT_MODE_CLAMP_TO_BORDER.

SamplerBorderColor **SAMPLER_BORDER_COLOR_INT_TRANSPARENT_BLACK** = `1`

Return a integer transparent black color when sampling outside the `[0.0, 1.0]` range. Only effective if the sampler repeat mode is SAMPLER_REPEAT_MODE_CLAMP_TO_BORDER.

SamplerBorderColor
SAMPLER_BORDER_COLOR_FLOAT_OPAQUE_BLACK = `2`

Return a floating-point opaque black color when sampling outside the `[0.0, 1.0]` range. Only effective if the sampler repeat mode is SAMPLER_REPEAT_MODE_CLAMP_TO_BORDER.

SamplerBorderColor
SAMPLER_BORDER_COLOR_INT_OPAQUE_BLACK = `3`

Return a integer opaque black color when sampling outside the `[0.0, 1.0]` range. Only effective if the sampler repeat mode is SAMPLER_REPEAT_MODE_CLAMP_TO_BORDER.

SamplerBorderColor
SAMPLER_BORDER_COLOR_FLOAT_OPAQUE_WHITE = `4`

Return a floating-point opaque white color when sampling outside the `[0.0, 1.0]` range. Only effective if the sampler repeat mode is SAMPLER_REPEAT_MODE_CLAMP_TO_BORDER.

SamplerBorderColor
SAMPLER_BORDER_COLOR_INT_OPAQUE_WHITE = `5`

Return a integer opaque white color when sampling outside the `[0.0, 1.0]` range. Only effective if the sampler repeat mode is SAMPLER_REPEAT_MODE_CLAMP_TO_BORDER.

SamplerBorderColor **SAMPLER_BORDER_COLOR_MAX** = `6`

Represents the size of the SamplerBorderColor enum.

---

enum **VertexFrequency**: 🔗

VertexFrequency **VERTEX_FREQUENCY_VERTEX** = `0`

Vertex attribute addressing is a function of the vertex. This is used to specify the rate at which vertex attributes are pulled from buffers.

VertexFrequency **VERTEX_FREQUENCY_INSTANCE** = `1`

Vertex attribute addressing is a function of the instance index. This is used to specify the rate at which vertex attributes are pulled from buffers.

---

enum **IndexBufferFormat**:

IndexBufferFormat **INDEX_BUFFER_FORMAT_UINT16** = `0`

Index buffer in 16-bit unsigned integer format. This limits the maximum index that can be specified to `65535`.

IndexBufferFormat **INDEX_BUFFER_FORMAT_UINT32** = `1`

Index buffer in 32-bit unsigned integer format. This limits the maximum index that can be specified to `4294967295`.

---

flags **StorageBufferUsage**:

StorageBufferUsage **STORAGE_BUFFER_USAGE_DISPATCH_INDIRECT** = `1`

There is currently no description for this enum. Please help us by contributing one!

---

enum **UniformType**:

UniformType **UNIFORM_TYPE_SAMPLER** = `0`

Sampler uniform.

UniformType **UNIFORM_TYPE_SAMPLER_WITH_TEXTURE** = `1`

Sampler uniform with a texture.

UniformType **UNIFORM_TYPE_TEXTURE** = `2`

Texture uniform.

UniformType **UNIFORM_TYPE_IMAGE** = `3`

Image uniform.

UniformType **UNIFORM_TYPE_TEXTURE_BUFFER** = `4`

Texture buffer uniform.

UniformType
UNIFORM_TYPE_SAMPLER_WITH_TEXTURE_BUFFER = `5`

Sampler uniform with a texture buffer.

UniformType **UNIFORM_TYPE_IMAGE_BUFFER** = `6`

Image buffer uniform.

UniformType **UNIFORM_TYPE_UNIFORM_BUFFER** = `7`

Uniform buffer uniform.

UniformType **UNIFORM_TYPE_STORAGE_BUFFER** = `8`

Storage buffer [https://vkguide.dev/docs/chapter-4/storage_buffers/] uniform.

UniformType **UNIFORM_TYPE_INPUT_ATTACHMENT** = `9`

Input attachment uniform.

UniformType **UNIFORM_TYPE_MAX** = `10`

Represents the size of the UniformType enum.

---

enum **RenderPrimitive**: 🔗

RenderPrimitive **RENDER_PRIMITIVE_POINTS** = `0`

Point rendering primitive (with constant size, regardless of distance from camera).

RenderPrimitive **RENDER_PRIMITIVE_LINES** = `1`

Line list rendering primitive. Lines are drawn separated from each other.

RenderPrimitive **RENDER_PRIMITIVE_LINES_WITH_ADJACENCY** = `2`

Line list rendering primitive with adjacency. [https://registry.khronos.org/vulkan/specs/1.3-extensions/html/vkspec.html#drawing-line-lists-with-adjacency]

Note: Adjacency is only useful with geometry shaders, which Godot does not expose.

RenderPrimitive **RENDER_PRIMITIVE_LINESTRIPS** = `3`

Line strip rendering primitive. Lines drawn are connected to the previous vertex.

RenderPrimitive **RENDER_PRIMITIVE_LINESTRIPS_WITH_ADJACENCY** = `4`

Line strip rendering primitive with adjacency. [https://registry.khronos.org/vulkan/specs/1.3-extensions/html/vkspec.html#drawing-line-strips-with-adjacency]

Note: Adjacency is only useful with geometry shaders, which Godot does not expose.

RenderPrimitive **RENDER_PRIMITIVE_TRIANGLES** = `5`

Triangle list rendering primitive. Triangles are drawn separated from each other.

RenderPrimitive **RENDER_PRIMITIVE_TRIANGLES_WITH_ADJACENCY** = `6`

Triangle list rendering primitive with adjacency. [https://registry.khronos.org/vulkan/specs/1.3-extensions/html/vkspec.html#drawing-triangle-lists-with-adjacency]

> **Note:** Adjacency is only useful with geometry shaders, which Godot does not expose.

RenderPrimitive **RENDER_PRIMITIVE_TRIANGLE_STRIPS** = `7`

Triangle strip rendering primitive. Triangles drawn are connected to the previous triangle.

RenderPrimitive **RENDER_PRIMITIVE_TRIANGLE_STRIPS_WITH_AJACENCY** = `8`

Triangle strip rendering primitive with adjacency. [https://registry.khronos.org/vulkan/specs/1.3-extensions/html/vkspec.html#drawing-triangle-strips-with-adjacency]

Note: Adjacency is only useful with geometry shaders, which Godot does not expose.

RenderPrimitive **RENDER_PRIMITIVE_TRIANGLE_STRIPS_WITH_RESTART_INDEX** = `9`

Triangle strip rendering primitive with *primitive restart* enabled. Triangles drawn are connected to the previous triangle, but a primitive restart index can be specified before drawing to create a second triangle strip after the specified index.

Note: Only compatible with indexed draws.

RenderPrimitive **RENDER_PRIMITIVE_TESSELATION_PATCH** = `10`

Tessellation patch rendering primitive. Only useful with tessellation shaders, which can be used to deform these patches.

RenderPrimitive **RENDER_PRIMITIVE_MAX** = `11`

Represents the size of the RenderPrimitive enum.

enum **PolygonCullMode**: 🔗

PolygonCullMode **POLYGON_CULL_DISABLED** = `0`

Do not use polygon front face or backface culling.

PolygonCullMode **POLYGON_CULL_FRONT** = `1`

Use polygon frontface culling (faces pointing towards the camera are hidden).

PolygonCullMode **POLYGON_CULL_BACK** = `2`

Use polygon backface culling (faces pointing away from the camera are hidden).

enum **PolygonFrontFace**: 🔗

PolygonFrontFace **POLYGON_FRONT_FACE_CLOCKWISE** = `0`

Clockwise winding order to determine which face of a polygon is its front face.

PolygonFrontFace
POLYGON_FRONT_FACE_COUNTER_CLOCKWISE = `1`

Counter-clockwise winding order to determine which face of a polygon is its front face.

enum **StencilOperation**: 🔗

StencilOperation **STENCIL_OP_KEEP** = `0`

Keep the current stencil value.

StencilOperation **STENCIL_OP_ZERO** = `1`

Set the stencil value to `0`.

StencilOperation **STENCIL_OP_REPLACE** = `2`

Replace the existing stencil value with the new one.

StencilOperation **STENCIL_OP_INCREMENT_AND_CLAMP** = `3`

Increment the existing stencil value and clamp to the maximum representable unsigned value if reached. Stencil bits are considered as an unsigned integer.

StencilOperation **STENCIL_OP_DECREMENT_AND_CLAMP** = `4`

Decrement the existing stencil value and clamp to the minimum value if reached. Stencil bits are considered as an unsigned integer.

StencilOperation **STENCIL_OP_INVERT** = `5`

Bitwise-invert the existing stencil value.

StencilOperation **STENCIL_OP_INCREMENT_AND_WRAP** = `6`

Increment the stencil value and wrap around to `0` if reaching the maximum representable unsigned. Stencil bits are considered as an unsigned integer.

StencilOperation **STENCIL_OP_DECREMENT_AND_WRAP** = `7`

Decrement the stencil value and wrap around to the maximum representable unsigned if reaching the minimum. Stencil bits are considered as an unsigned integer.

StencilOperation **STENCIL_OP_MAX** = `8`

Represents the size of the StencilOperation enum.

enum **CompareOperator**: 🔗

CompareOperator **COMPARE_OP_NEVER** = `0`

"Never" comparison (opposite of COMPARE_OP_ALWAYS).

CompareOperator **COMPARE_OP_LESS** = `1`

"Less than" comparison.

CompareOperator **COMPARE_OP_EQUAL** = `2`

"Equal" comparison.

CompareOperator **COMPARE_OP_LESS_OR_EQUAL** = `3`

"Less than or equal" comparison.

CompareOperator **COMPARE_OP_GREATER** = `4`

"Greater than" comparison.

CompareOperator **COMPARE_OP_NOT_EQUAL** = `5`

"Not equal" comparison.

CompareOperator **COMPARE_OP_GREATER_OR_EQUAL** = `6`

"Greater than or equal" comparison.

CompareOperator **COMPARE_OP_ALWAYS** = `7`

"Always" comparison (opposite of COMPARE_OP_NEVER).

CompareOperator **COMPARE_OP_MAX** = `8`

Represents the size of the CompareOperator enum.

enum **LogicOperation**: 🔗

LogicOperation **LOGIC_OP_CLEAR** = `0`

Clear logic operation (result is always `0`). See also LOGIC_OP_SET.

LogicOperation **LOGIC_OP_AND** = `1`

AND logic operation.

LogicOperation **LOGIC_OP_AND_REVERSE** = `2`

AND logic operation with the *destination* operand being inverted. See also LOGIC_OP_AND_INVERTED.

LogicOperation **LOGIC_OP_COPY** = `3`

Copy logic operation (keeps the *source* value as-is). See also LOGIC_OP_COPY_INVERTED and LOGIC_OP_NO_OP.

LogicOperation **LOGIC_OP_AND_INVERTED** = `4`

AND logic operation with the *source* operand being inverted. See also LOGIC_OP_AND_REVERSE.

LogicOperation **LOGIC_OP_NO_OP** = `5`

No-op logic operation (keeps the *destination* value as-is). See also LOGIC_OP_COPY.

LogicOperation **LOGIC_OP_XOR** = `6`

Exclusive or (XOR) logic operation.

LogicOperation **LOGIC_OP_OR** = `7`

OR logic operation.

LogicOperation **LOGIC_OP_NOR** = `8`

Not-OR (NOR) logic operation.

LogicOperation **LOGIC_OP_EQUIVALENT** = `9`

Not-XOR (XNOR) logic operation.

LogicOperation **LOGIC_OP_INVERT** = `10`

Invert logic operation.

LogicOperation **LOGIC_OP_OR_REVERSE** = `11`

OR logic operation with the *destination* operand being inverted. See also LOGIC_OP_OR_REVERSE.

LogicOperation **LOGIC_OP_COPY_INVERTED** = `12`

NOT logic operation (inverts the value). See also LOGIC_OP_COPY.

LogicOperation **LOGIC_OP_OR_INVERTED** = `13`

OR logic operation with the *source* operand being inverted. See also LOGIC_OP_OR_REVERSE.

LogicOperation **LOGIC_OP_NAND** = `14`

Not-AND (NAND) logic operation.

LogicOperation **LOGIC_OP_SET** = `15`

SET logic operation (result is always `1`). See also LOGIC_OP_CLEAR.

LogicOperation **LOGIC_OP_MAX** = `16`

Represents the size of the LogicOperation enum.

---

enum **BlendFactor**: 🔗

BlendFactor **BLEND_FACTOR_ZERO** = `0`

Constant `0.0` blend factor.

BlendFactor **BLEND_FACTOR_ONE** = `1`

Constant `1.0` blend factor.

BlendFactor **BLEND_FACTOR_SRC_COLOR** = `2`

Color blend factor is `source color`. Alpha blend factor is `source alpha`.

BlendFactor **BLEND_FACTOR_ONE_MINUS_SRC_COLOR** = `3`

Color blend factor is `1.0 - source color`. Alpha blend factor is `1.0 - source alpha`.

BlendFactor **BLEND_FACTOR_DST_COLOR** = `4`

Color blend factor is `destination color`. Alpha blend factor is `destination alpha`.

BlendFactor **BLEND_FACTOR_ONE_MINUS_DST_COLOR** = `5`

Color blend factor is `1.0 - destination color`. Alpha blend factor is `1.0 - destination alpha`.

BlendFactor **BLEND_FACTOR_SRC_ALPHA** = `6`

Color and alpha blend factor is `source alpha`.

BlendFactor **BLEND_FACTOR_ONE_MINUS_SRC_ALPHA** = `7`

Color and alpha blend factor is `1.0 - source alpha`.

BlendFactor **BLEND_FACTOR_DST_ALPHA** = `8`

Color and alpha blend factor is `destination alpha`.

BlendFactor **BLEND_FACTOR_ONE_MINUS_DST_ALPHA** = `9`

Color and alpha blend factor is `1.0 - destination alpha`.

BlendFactor **BLEND_FACTOR_CONSTANT_COLOR** = `10`

Color blend factor is `blend constant color`. Alpha blend factor is `blend constant alpha` (see draw_list_set_blend_constants).

BlendFactor **BLEND_FACTOR_ONE_MINUS_CONSTANT_COLOR** = `11`

Color blend factor is `1.0 - blend constant color`. Alpha blend factor is `1.0 - blend constant alpha` (see draw_list_set_blend_constants).

BlendFactor **BLEND_FACTOR_CONSTANT_ALPHA** = `12`

Color and alpha blend factor is `blend constant alpha` (see draw_list_set_blend_constants).

BlendFactor **BLEND_FACTOR_ONE_MINUS_CONSTANT_ALPHA** = `13`

Color and alpha blend factor is `1.0 - blend constant alpha` (see draw_list_set_blend_constants).

BlendFactor **BLEND_FACTOR_SRC_ALPHA_SATURATE** = `14`

Color blend factor is `min(source alpha, 1.0 - destination alpha)`. Alpha blend factor is `1.0`.

BlendFactor **BLEND_FACTOR_SRC1_COLOR** = `15`

Color blend factor is `second source color`. Alpha blend factor is `second source alpha`. Only relevant for dual-source blending.

BlendFactor **BLEND_FACTOR_ONE_MINUS_SRC1_COLOR** = `16`

Color blend factor is `1.0 - second source color`. Alpha blend factor is `1.0 - second source alpha`. Only relevant for dual-source blending.

BlendFactor **BLEND_FACTOR_SRC1_ALPHA** = `17`

Color and alpha blend factor is `second source alpha`. Only relevant for dual-source blending.

BlendFactor **BLEND_FACTOR_ONE_MINUS_SRC1_ALPHA** = `18`

Color and alpha blend factor is `1.0 - second source alpha`. Only relevant for dual-source blending.

BlendFactor **BLEND_FACTOR_MAX** = `19`

Represents the size of the BlendFactor enum.

enum **BlendOperation**: 🔗

BlendOperation **BLEND_OP_ADD** = `0`

Additive blending operation (`source + destination`).

BlendOperation **BLEND_OP_SUBTRACT** = `1`

Subtractive blending operation (`source - destination`).

BlendOperation **BLEND_OP_REVERSE_SUBTRACT** = `2`

Reverse subtractive blending operation (`destination - source`).

BlendOperation **BLEND_OP_MINIMUM** = `3`

Minimum blending operation (keep the lowest value of the two).

BlendOperation **BLEND_OP_MAXIMUM** = `4`

Maximum blending operation (keep the highest value of the two).

BlendOperation **BLEND_OP_MAX** = `5`

Represents the size of the BlendOperation enum.

---

flags **PipelineDynamicStateFlags**: 🔗

PipelineDynamicStateFlags **DYNAMIC_STATE_LINE_WIDTH** = `1`

Allows dynamically changing the width of rendering lines.

PipelineDynamicStateFlags **DYNAMIC_STATE_DEPTH_BIAS** = `2`

Allows dynamically changing the depth bias.

PipelineDynamicStateFlags
DYNAMIC_STATE_BLEND_CONSTANTS = `4`

There is currently no description for this enum. Please help us by contributing one!

PipelineDynamicStateFlags **DYNAMIC_STATE_DEPTH_BOUNDS** = `8`

There is currently no description for this enum. Please help us by contributing one!

PipelineDynamicStateFlags **DYNAMIC_STATE_STENCIL_COMPARE_MASK** = `16`

There is currently no description for this enum. Please help us by contributing one!

PipelineDynamicStateFlags **DYNAMIC_STATE_STENCIL_WRITE_MASK** = `32`

There is currently no description for this enum. Please help us by contributing one!

PipelineDynamicStateFlags **DYNAMIC_STATE_STENCIL_REFERENCE** = `64`

There is currently no description for this enum. Please help us by contributing one!

---

enum **InitialAction**: 🔗

InitialAction **INITIAL_ACTION_LOAD** = `0`

Load the previous contents of the framebuffer.

InitialAction **INITIAL_ACTION_CLEAR** = `1`

Clear the whole framebuffer or its specified region.

InitialAction **INITIAL_ACTION_DISCARD** = `2`

Ignore the previous contents of the framebuffer. This is the fastest option if you'll overwrite all of the pixels and don't need to read any of them.

InitialAction **INITIAL_ACTION_MAX** = `3`

Represents the size of the InitialAction enum.

InitialAction **INITIAL_ACTION_CLEAR_REGION** = `1`

Deprecated: Use INITIAL_ACTION_CLEAR instead.

InitialAction **INITIAL_ACTION_CLEAR_REGION_CONTINUE** = `1`

Deprecated: Use INITIAL_ACTION_LOAD instead.

InitialAction **INITIAL_ACTION_KEEP** = `0`

Deprecated: Use INITIAL_ACTION_LOAD instead.

InitialAction **INITIAL_ACTION_DROP** = `2`

Deprecated: Use INITIAL_ACTION_DISCARD instead.

InitialAction **INITIAL_ACTION_CONTINUE** = `0`

Deprecated: Use INITIAL_ACTION_LOAD instead.

---

enum **FinalAction**: 🔗

FinalAction **FINAL_ACTION_STORE** = `0`

Store the result of the draw list in the framebuffer. This is generally what you want to do.

FinalAction **FINAL_ACTION_DISCARD** = `1`

Discard the contents of the framebuffer. This is the fastest option if you don't need to use the results of the draw list.

FinalAction **FINAL_ACTION_MAX** = `2`

Represents the size of the FinalAction enum.

FinalAction **FINAL_ACTION_READ** = `0`

Deprecated: Use FINAL_ACTION_STORE instead.

FinalAction **FINAL_ACTION_CONTINUE** = `0`

Deprecated: Use FINAL_ACTION_STORE instead.

---

enum **ShaderStage**: 🔗

ShaderStage **SHADER_STAGE_VERTEX** = `0`

Vertex shader stage. This can be used to manipulate vertices from a shader (but not create new vertices).

ShaderStage **SHADER_STAGE_FRAGMENT** = `1`

Fragment shader stage (called "pixel shader" in Direct3D). This can be used to manipulate pixels from a shader.

ShaderStage **SHADER_STAGE_TESSELATION_CONTROL** = `2`

Tessellation control shader stage. This can be used to create additional geometry from a shader.

ShaderStage **SHADER_STAGE_TESSELATION_EVALUATION** = `3`

Tessellation evaluation shader stage. This can be used to create additional geometry from a shader.

ShaderStage **SHADER_STAGE_COMPUTE** = `4`

Compute shader stage. This can be used to run arbitrary computing tasks in a shader, performing them on the GPU instead of the CPU.

ShaderStage **SHADER_STAGE_MAX** = `5`

Represents the size of the ShaderStage enum.

ShaderStage **SHADER_STAGE_VERTEX_BIT** = `1`

Vertex shader stage bit (see also SHADER_STAGE_VERTEX).

ShaderStage **SHADER_STAGE_FRAGMENT_BIT** = `2`

Fragment shader stage bit (see also SHADER_STAGE_FRAGMENT).

ShaderStage **SHADER_STAGE_TESSELATION_CONTROL_BIT** = `4`

Tessellation control shader stage bit (see also SHADER_STAGE_TESSELATION_CONTROL).

ShaderStage **SHADER_STAGE_TESSELATION_EVALUATION_BIT** = `8`

Tessellation evaluation shader stage bit (see also SHADER_STAGE_TESSELATION_EVALUATION).

ShaderStage **SHADER_STAGE_COMPUTE_BIT** = `16`

Compute shader stage bit (see also SHADER_STAGE_COMPUTE).

enum **ShaderLanguage**: 🔗

ShaderLanguage **SHADER_LANGUAGE_GLSL** = `0`

Khronos' GLSL shading language (used natively by OpenGL and Vulkan). This is the language used for core Godot shaders.

ShaderLanguage **SHADER_LANGUAGE_HLSL** = `1`

Microsoft's High-Level Shading Language (used natively by Direct3D, but can also be used in Vulkan).

enum **PipelineSpecializationConstantType**:

PipelineSpecializationConstantType **PIPELINE_SPECIALIZATION_CONSTANT_TYPE_BOOL** = `0`

Boolean specialization constant.

PipelineSpecializationConstantType **PIPELINE_SPECIALIZATION_CONSTANT_TYPE_INT** = `1`

Integer specialization constant.

PipelineSpecializationConstantType **PIPELINE_SPECIALIZATION_CONSTANT_TYPE_FLOAT** = `2`

Floating-point specialization constant.

enum **Limit**:

Limit **LIMIT_MAX_BOUND_UNIFORM_SETS** = `0`

Maximum number of uniform sets that can be bound at a given time.

Limit **LIMIT_MAX_FRAMEBUFFER_COLOR_ATTACHMENTS** = `1`

Maximum number of color framebuffer attachments that can be used at a given time.

Limit **LIMIT_MAX_TEXTURES_PER_UNIFORM_SET** = `2`

Maximum number of textures that can be used per uniform set.

Limit **LIMIT_MAX_SAMPLERS_PER_UNIFORM_SET** = `3`

Maximum number of samplers that can be used per uniform set.

Limit **LIMIT_MAX_STORAGE_BUFFERS_PER_UNIFORM_SET** = `4`

Maximum number of storage buffers [https://vkguide.dev/docs/chapter-4/storage_buffers/] per uniform set.

Limit **LIMIT_MAX_STORAGE_IMAGES_PER_UNIFORM_SET** = `5`

Maximum number of storage images per uniform set.

Limit **LIMIT_MAX_UNIFORM_BUFFERS_PER_UNIFORM_SET** = `6`

Maximum number of uniform buffers per uniform set.

Limit **LIMIT_MAX_DRAW_INDEXED_INDEX** = `7`

Maximum index for an indexed draw command.

Limit **LIMIT_MAX_FRAMEBUFFER_HEIGHT** = `8`

Maximum height of a framebuffer (in pixels).

Limit **LIMIT_MAX_FRAMEBUFFER_WIDTH** = `9`

Maximum width of a framebuffer (in pixels).

Limit **LIMIT_MAX_TEXTURE_ARRAY_LAYERS** = `10`

Maximum number of texture array layers.

Limit **LIMIT_MAX_TEXTURE_SIZE_1D** = `11`

Maximum supported 1-dimensional texture size (in pixels on a single axis).

Limit **LIMIT_MAX_TEXTURE_SIZE_2D** = `12`

Maximum supported 2-dimensional texture size (in pixels on a single axis).

Limit **LIMIT_MAX_TEXTURE_SIZE_3D** = `13`

Maximum supported 3-dimensional texture size (in pixels on a single axis).

Limit **LIMIT_MAX_TEXTURE_SIZE_CUBE** = `14`

Maximum supported cubemap texture size (in pixels on a single axis of a single face).

Limit **LIMIT_MAX_TEXTURES_PER_SHADER_STAGE** = `15`

Maximum number of textures per shader stage.

Limit **LIMIT_MAX_SAMPLERS_PER_SHADER_STAGE** = `16`

Maximum number of samplers per shader stage.

Limit **LIMIT_MAX_STORAGE_BUFFERS_PER_SHADER_STAGE** = `17`

Maximum number of storage buffers [https://vkguide.dev/docs/chapter-4/storage_buffers/] per shader stage.

Limit **LIMIT_MAX_STORAGE_IMAGES_PER_SHADER_STAGE** = `18`

Maximum number of storage images per shader stage.

Limit **LIMIT_MAX_UNIFORM_BUFFERS_PER_SHADER_STAGE** = `19`

Maximum number of uniform buffers per uniform set.

Limit **LIMIT_MAX_PUSH_CONSTANT_SIZE** = `20`

Maximum size of a push constant. A lot of devices are limited to 128 bytes, so try to avoid exceeding 128 bytes in push constants to ensure compatibility even if your GPU is reporting a higher value.

Limit **LIMIT_MAX_UNIFORM_BUFFER_SIZE** = `21`

Maximum size of a uniform buffer.

Limit **LIMIT_MAX_VERTEX_INPUT_ATTRIBUTE_OFFSET** = `22`

Maximum vertex input attribute offset.

Limit **LIMIT_MAX_VERTEX_INPUT_ATTRIBUTES** = `23`

Maximum number of vertex input attributes.

Limit **LIMIT_MAX_VERTEX_INPUT_BINDINGS** = `24`

Maximum number of vertex input bindings.

Limit **LIMIT_MAX_VERTEX_INPUT_BINDING_STRIDE** = `25`

Maximum vertex input binding stride.

Limit **LIMIT_MIN_UNIFORM_BUFFER_OFFSET_ALIGNMENT** = `26`

Minimum uniform buffer offset alignment.

Limit **LIMIT_MAX_COMPUTE_SHARED_MEMORY_SIZE** = `27`

Maximum shared memory size for compute shaders.

Limit **LIMIT_MAX_COMPUTE_WORKGROUP_COUNT_X** = `28`

Maximum number of workgroups for compute shaders on the X axis.

Limit **LIMIT_MAX_COMPUTE_WORKGROUP_COUNT_Y** = `29`

Maximum number of workgroups for compute shaders on the Y axis.

Limit **LIMIT_MAX_COMPUTE_WORKGROUP_COUNT_Z** = `30`

Maximum number of workgroups for compute shaders on the Z axis.

Limit **LIMIT_MAX_COMPUTE_WORKGROUP_INVOCATIONS** = `31`

Maximum number of workgroup invocations for compute shaders.

Limit **LIMIT_MAX_COMPUTE_WORKGROUP_SIZE_X** = `32`

Maximum workgroup size for compute shaders on the X axis.

Limit **LIMIT_MAX_COMPUTE_WORKGROUP_SIZE_Y** = `33`

Maximum workgroup size for compute shaders on the Y axis.

Limit **LIMIT_MAX_COMPUTE_WORKGROUP_SIZE_Z** = `34`

Maximum workgroup size for compute shaders on the Z axis.

Limit **LIMIT_MAX_VIEWPORT_DIMENSIONS_X** = `35`

Maximum viewport width (in pixels).

Limit **LIMIT_MAX_VIEWPORT_DIMENSIONS_Y** = `36`

Maximum viewport height (in pixels).

---

enum **MemoryType**: 🔗

MemoryType **MEMORY_TEXTURES** = `0`

Memory taken by textures.

MemoryType **MEMORY_BUFFERS** = `1`

Memory taken by buffers.

MemoryType **MEMORY_TOTAL** = `2`

Total memory taken. This is greater than the sum of MEMORY_TEXTURES and MEMORY_BUFFERS, as it also includes miscellaneous memory usage.

Constants

INVALID_ID = `-1` 🔗

Returned by functions that return an ID if a value is invalid.

INVALID_FORMAT_ID = `-1` 🔗

Returned by functions that return a format ID if a value is invalid.

Method Descriptions

void **barrier**(from: BitField[BarrierMask] = 32767, to: BitField[BarrierMask] = 32767) 🔗

Deprecated: Barriers are automatically inserted by RenderingDevice.

This method does nothing.

Error **buffer_clear**(buffer: RID, offset: int, size_bytes: int) 🔗

Clears the contents of the `buffer`, clearing `size_bytes` bytes, starting at `offset`.

Prints an error if:

- the size isn't a multiple of four
- the region specified by `offset` + `size_bytes` exceeds the buffer
- a draw list is currently active (created by draw_list_begin)
- a compute list is currently active (created by compute_list_begin)

Error **buffer_copy**(src_buffer: RID, dst_buffer: RID, src_offset: int, dst_offset: int, size: int) 🔗

Copies `size` bytes from the `src_buffer` at `src_offset` into `dst_buffer` at `dst_offset`.

Prints an error if:

- `size` exceeds the size of either `src_buffer` or `dst_buffer` at their corresponding offsets
- a draw list is currently active (created by draw_list_begin)
- a compute list is currently active (created by compute_list_begin)

PackedByteArray **buffer_get_data**(buffer: RID, offset_bytes: int = 0, size_bytes: int = 0) 🔗

Returns a copy of the data of the specified `buffer`, optionally `offset_bytes` and `size_bytes` can be set to copy only a portion of the buffer.

Error **buffer_update**(buffer: RID, offset: int, size_bytes: int, data: PackedByteArray) 🔗

Updates a region of `size_bytes` bytes, starting at `offset`, in the buffer, with the specified `data`.

Prints an error if:

- the region specified by `offset` + `size_bytes` exceeds the buffer
- a draw list is currently active (created by draw_list_begin)

- a compute list is currently active (created by compute_list_begin)

---

void **capture_timestamp**(name: String)

Creates a timestamp marker with the specified `name`. This is used for performance reporting with the get_captured_timestamp_cpu_time, get_captured_timestamp_gpu_time and get_captured_timestamp_name methods.

---

void **compute_list_add_barrier**(compute_list: int)

Raises a Vulkan compute barrier in the specified `compute_list`.

---

int **compute_list_begin**()

Starts a list of compute commands created with the `compute_*` methods. The returned value should be passed to other `compute_list_*` functions.

Multiple compute lists cannot be created at the same time; you must finish the previous compute list first using compute_list_end.

A simple compute operation might look like this (code is not a complete example):

```
var rd = RenderingDevice.new()
var compute_list = rd.compute_list_begin()

rd.compute_list_bind_compute_pipeline(compute_list,
compute_shader_dilate_pipeline)
rd.compute_list_bind_uniform_set(compute_list,
compute_base_uniform_set, 0)
rd.compute_list_bind_uniform_set(compute_list,
dilate_uniform_set, 1)

for i in atlas_slices:
    rd.compute_list_set_push_constant(compute_list,
push_constant, push_constant.size())
    rd.compute_list_dispatch(compute_list, group_size.x,
group_size.y, group_size.z)
    # No barrier, let them run all together.

rd.compute_list_end()
```

void **compute_list_bind_compute_pipeline**(compute_list: int, compute_pipeline: RID) 🔗

Tells the GPU what compute pipeline to use when processing the compute list. If the shader has changed since the last time this function was called, Godot will unbind all descriptor sets and will re-bind them inside compute_list_dispatch.

void **compute_list_bind_uniform_set**(compute_list: int, uniform_set: RID, set_index: int) 🔗

Binds the `uniform_set` to this `compute_list`. Godot ensures that all textures in the uniform set have the correct Vulkan access masks. If Godot had to change access masks of textures, it will raise a Vulkan image memory barrier.

void **compute_list_dispatch**(compute_list: int, x_groups: int, y_groups: int, z_groups: int) 🔗

Submits the compute list for processing on the GPU. This is the compute equivalent to draw_list_draw.

void **compute_list_dispatch_indirect**(compute_list: int, buffer: RID, offset: int) 🔗

Submits the compute list for processing on the GPU with the given group counts stored in the `buffer` at `offset`. Buffer must have been created with STORAGE_BUFFER_USAGE_DISPATCH_INDIRECT flag.

---

void **compute_list_end**() 🔗

Finishes a list of compute commands created with the `compute_*` methods.

---

void **compute_list_set_push_constant**(compute_list: int, buffer: PackedByteArray, size_bytes: int) 🔗

Sets the push constant data to `buffer` for the specified `compute_list`. The shader determines how this binary data is used. The buffer's size in bytes must also be specified in `size_bytes` (this can be obtained by calling the PackedByteArray.size method on the passed `buffer`).

RID **compute_pipeline_create**(shader: RID, specialization_constants: Array[RDPipelineSpecializationConstant] = []) 🔗

Creates a new compute pipeline. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

---

bool **compute_pipeline_is_valid**(compute_pipeline: RID) 🔗

Returns `true` if the compute pipeline specified by the `compute_pipeline` RID is valid, `false` otherwise.

RenderingDevice **create_local_device**() 🔗

Create a new local **RenderingDevice**. This is most useful for performing compute operations on the GPU independently from the rest of the engine.

---

void **draw_command_begin_label**(name: String, color: Color)

Create a command buffer debug label region that can be displayed in third-party tools such as RenderDoc [https://renderdoc.org/]. All regions must be ended with a draw_command_end_label call. When viewed from the linear series of submissions to a single queue, calls to draw_command_begin_label and draw_command_end_label must be matched and balanced.

The `VK_EXT_DEBUG_UTILS_EXTENSION_NAME` Vulkan extension must be available and enabled for command buffer debug label region to work. See also draw_command_end_label.

---

void **draw_command_end_label**()

Ends the command buffer debug label region started by a draw_command_begin_label call.

---

void **draw_command_insert_label**(name: String, color: Color)

Deprecated: Inserting labels no longer applies due to command reordering.

This method does nothing.

---

int **draw_list_begin**(framebuffer: RID, initial_color_action: InitialAction, final_color_action: FinalAction, initial_depth_action: InitialAction, final_depth_action: FinalAction, clear_color_values: PackedColorArray = PackedColorArray(), clear_depth: float = 1.0, clear_stencil: int = 0, region: Rect2 = Rect2(0, 0, 0, 0))

Starts a list of raster drawing commands created with the `draw_*` methods. The returned value should be passed to other `draw_list_*` functions.

Multiple draw lists cannot be created at the same time; you must finish the previous draw list first using draw_list_end.

A simple drawing operation might look like this (code is not a complete example):

```
var rd = RenderingDevice.new()
var clear_colors = PackedColorArray([Color(0, 0, 0, 0), Color(0,
0, 0, 0), Color(0, 0, 0, 0)])
var draw_list = rd.draw_list_begin(framebuffers[i],
RenderingDevice.INITIAL_ACTION_CLEAR,
RenderingDevice.FINAL_ACTION_READ,
RenderingDevice.INITIAL_ACTION_CLEAR,
RenderingDevice.FINAL_ACTION_DISCARD, clear_colors)

# Draw opaque.
rd.draw_list_bind_render_pipeline(draw_list, raster_pipeline)
rd.draw_list_bind_uniform_set(draw_list, raster_base_uniform, 0)
rd.draw_list_set_push_constant(draw_list, raster_push_constant,
raster_push_constant.size())
rd.draw_list_draw(draw_list, false, 1, slice_triangle_count[i] *
3)
# Draw wire.
rd.draw_list_bind_render_pipeline(draw_list,
raster_pipeline_wire)
rd.draw_list_bind_uniform_set(draw_list, raster_base_uniform, 0)
rd.draw_list_set_push_constant(draw_list, raster_push_constant,
raster_push_constant.size())
rd.draw_list_draw(draw_list, false, 1, slice_triangle_count[i] *
3)

rd.draw_list_end()
```

---

int **draw_list_begin_for_screen**(screen: int = 0, clear_color: Color = Color(0, 0, 0, 1)) 🔗

High-level variant of draw_list_begin, with the parameters automatically being adjusted for drawing onto the window specified by the `screen` ID.

Note: Cannot be used with local RenderingDevices, as these don't have a screen. If called on a local RenderingDevice, draw_list_begin_for_screen returns INVALID_ID.

PackedInt64Array **draw_list_begin_split**(framebuffer: RID, splits: int, initial_color_action: InitialAction, final_color_action: FinalAction, initial_depth_action: InitialAction, final_depth_action: FinalAction, clear_color_values: PackedColorArray = PackedColorArray(), clear_depth: float = 1.0, clear_stencil: int = 0, region: Rect2 = Rect2(0, 0, 0, 0), storage_textures: Array[RID] = [])

Deprecated: Split draw lists are used automatically by RenderingDevice.

This method does nothing and always returns an empty PackedInt64Array.

---

void **draw_list_bind_index_array**(draw_list: int, index_array: RID)

Binds `index_array` to the specified `draw_list`.

---

void **draw_list_bind_render_pipeline**(draw_list: int, render_pipeline: RID)

Binds `render_pipeline` to the specified `draw_list`.

---

void **draw_list_bind_uniform_set**(draw_list: int, uniform_set: RID, set_index: int)

Binds `uniform_set` to the specified `draw_list`. A `set_index` must also be specified, which is an identifier starting from `0` that must match the one expected by the draw list.

---

void **draw_list_bind_vertex_array**(draw_list: int, vertex_array: RID) 🔗

Binds `vertex_array` to the specified `draw_list`.

---

void **draw_list_disable_scissor**(draw_list: int) 🔗

Removes and disables the scissor rectangle for the specified `draw_list`. See also draw_list_enable_scissor.

---

void **draw_list_draw**(draw_list: int, use_indices: bool, instances: int, procedural_vertex_count: int = 0) 🔗

Submits `draw_list` for rendering on the GPU. This is the raster equivalent to compute_list_dispatch.

---

void **draw_list_enable_scissor**(draw_list: int, rect: Rect2 = Rect2(0, 0, 0, 0)) 🔗

Creates a scissor rectangle and enables it for the specified `draw_list`. Scissor rectangles are used for clipping by discarding fragments that fall outside a specified rectangular portion of the screen. See also draw_list_disable_scissor.

Note: The specified `rect` is automatically intersected with the screen's dimensions, which means it cannot exceed the screen's dimensions.

---

void **draw_list_end**() 🔗

Finishes a list of raster drawing commands created with the `draw_*` methods.

---

void **draw_list_set_blend_constants**(draw_list: int, color: Color)

Sets blend constants for the specified `draw_list` to `color`. Blend constants are used only if the graphics pipeline is created with DYNAMIC_STATE_BLEND_CONSTANTS flag set.

---

void **draw_list_set_push_constant**(draw_list: int, buffer: PackedByteArray, size_bytes: int)

Sets the push constant data to `buffer` for the specified `draw_list`. The shader determines how this binary data is used. The buffer's size in bytes must also be specified in `size_bytes` (this can be obtained by calling the PackedByteArray.size method on the passed `buffer`).

---

int **draw_list_switch_to_next_pass**()

Switches to the next draw pass.

PackedInt64Array **draw_list_switch_to_next_pass_split**(splits: int)

Deprecated: Split draw lists are used automatically by RenderingDevice.

This method does nothing and always returns an empty PackedInt64Array.

RID **framebuffer_create**(textures: Array[RID], validate_with_format: int = -1, view_count: int = 1) 🔗

Creates a new framebuffer. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

RID **framebuffer_create_empty**(size: Vector2i, samples: TextureSamples = 0, validate_with_format: int = -1) 🔗

Creates a new empty framebuffer. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

RID **framebuffer_create_multipass**(textures: Array[RID], passes: Array[RDFramebufferPass], validate_with_format: int = -1, view_count: int = 1) 🔗

Creates a new multipass framebuffer. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

---

int **framebuffer_format_create**(attachments: Array[RDAttachmentFormat], view_count: int = 1) 🔗

Creates a new framebuffer format with the specified `attachments` and `view_count`. Returns the new framebuffer's unique framebuffer format ID.

If `view_count` is greater than or equal to `2`, enables multiview which is used for VR rendering. This requires support for the Vulkan multiview extension.

---

int **framebuffer_format_create_empty**(samples: TextureSamples = 0)

Creates a new empty framebuffer format with the specified number of `samples` and returns its ID.

---

int **framebuffer_format_create_multipass**(attachments: Array[RDAttachmentFormat], passes: Array[RDFramebufferPass], view_count: int = 1)

Creates a multipass framebuffer format with the specified `attachments`, `passes` and `view_count` and returns its ID. If `view_count` is greater than or equal to `2`, enables multiview which is used for VR rendering. This requires support for the Vulkan multiview extension.

TextureSamples **framebuffer_format_get_texture_samples**(format: int, render_pass: int = 0)

Returns the number of texture samples used for the given framebuffer `format` ID (returned by framebuffer_get_format).

---

int **framebuffer_get_format**(framebuffer: RID)

Returns the format ID of the framebuffer specified by the `framebuffer` RID. This ID is guaranteed to be unique for the same formats and does not need to be freed.

---

bool **framebuffer_is_valid**(framebuffer: RID) const

Returns `true` if the framebuffer specified by the `framebuffer` RID is valid, `false` otherwise.

---

void **free_rid**(rid: RID)

Tries to free an object in the RenderingDevice. To avoid memory leaks, this should be called after using an object as memory management does not occur automatically when using RenderingDevice directly.

---

void **full_barrier**()

Deprecated: Barriers are automatically inserted by RenderingDevice.

This method does nothing.

---

int **get_captured_timestamp_cpu_time**(index: int) const

Returns the timestamp in CPU time for the rendering step specified by `index` (in microseconds since the engine started). See also get_captured_timestamp_gpu_time and capture_timestamp.

---

int **get_captured_timestamp_gpu_time**(index: int) const

Returns the timestamp in GPU time for the rendering step specified by `index` (in microseconds since the engine started). See also get_captured_timestamp_cpu_time and capture_timestamp.

String **get_captured_timestamp_name**(index: int) const

Returns the timestamp's name for the rendering step specified by `index`. See also capture_timestamp.

---

int **get_captured_timestamps_count**() const 🔗

Returns the total number of timestamps (rendering steps) available for profiling.

---

int **get_captured_timestamps_frame**() const 🔗

Returns the index of the last frame rendered that has rendering timestamps available for querying.

String **get_device_name**() const 🔗

Returns the name of the video adapter (e.g. "GeForce GTX 1080/PCIe/SSE2"). Equivalent to RenderingServer.get_video_adapter_name. See also get_device_vendor_name.

String **get_device_pipeline_cache_uuid**() const 🔗

Returns the universally unique identifier for the pipeline cache. This is used to cache shader files on disk, which avoids shader recompilations on subsequent engine runs. This UUID varies depending on the graphics card model, but also the driver version. Therefore, updating graphics drivers will invalidate the shader cache.

String **get_device_vendor_name**() const 🔗

Returns the vendor of the video adapter (e.g. "NVIDIA Corporation"). Equivalent to RenderingServer.get_video_adapter_vendor. See also get_device_name.

---

int **get_driver_resource**(resource: DriverResource, rid: RID, index: int)

Returns the unique identifier of the driver `resource` for the specified `rid`. Some driver resource types ignore the specified `rid` (see DriverResource descriptions). `index` is always ignored but must be specified anyway.

---

int **get_frame_delay**() const

Returns the frame count kept by the graphics API. Higher values result in higher input lag, but with more consistent throughput. For the main **RenderingDevice**, frames are cycled (usually 3 with triple-buffered V-Sync enabled). However, local **RenderingDevice**s only have 1 frame.

---

int **get_memory_usage**(type: MemoryType) const

Returns the memory usage in bytes corresponding to the given `type`. When using Vulkan, these statistics are calculated by Vulkan Memory Allocator [https://github.com/GPUOpen-LibrariesAndSDKs/VulkanMemoryAllocator].

RID **index_array_create**(index_buffer: RID, index_offset: int, index_count: int)

Creates a new index array. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

RID **index_buffer_create**(size_indices: int, format: IndexBufferFormat, data: PackedByteArray = PackedByteArray(), use_restart_indices: bool = false)

Creates a new index buffer. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

---

int **limit_get**(limit: Limit) const

Returns the value of the specified `limit`. This limit varies depending on the current graphics hardware (and sometimes the driver version). If the given limit is exceeded, rendering errors will occur.

Limits for various graphics hardware can be found in the Vulkan Hardware Database [https://vulkan.gpuinfo.org/].

RID **render_pipeline_create**(shader: RID, framebuffer_format: int, vertex_format: int, primitive: RenderPrimitive, rasterization_state: RDPipelineRasterizationState, multisample_state: RDPipelineMultisampleState, stencil_state: RDPipelineDepthStencilState, color_blend_state: RDPipelineColorBlendState, dynamic_state_flags: BitField[PipelineDynamicStateFlags] = 0, for_render_pass: int = 0, specialization_constants: Array[RDPipelineSpecializationConstant] = [])

Creates a new render pipeline. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

---

bool **render_pipeline_is_valid**(render_pipeline: RID)

Returns `true` if the render pipeline specified by the `render_pipeline` RID is valid, `false` otherwise.

RID **sampler_create**(state: RDSamplerState) 🔗

Creates a new sampler. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

---

bool **sampler_is_format_supported_for_filter**(format: DataFormat, sampler_filter: SamplerFilter) const 🔗

Returns `true` if implementation supports using a texture of `format` with the given `sampler_filter`.

---

int **screen_get_framebuffer_format**(screen: int = 0) const 🔗

Returns the framebuffer format of the given screen.

Note: Only the main **RenderingDevice** returned by RenderingServer.get_rendering_device has a format. If called on a local **RenderingDevice**, this method prints an error and returns INVALID_ID.

---

int **screen_get_height**(screen: int = 0) const 🔗

Returns the window height matching the graphics API context for the given window ID (in pixels). Despite the parameter being named `screen`, this returns the *window* size. See also screen_get_width.

Note: Only the main **RenderingDevice** returned by RenderingServer.get_rendering_device has a height. If called on a local **RenderingDevice**, this method prints an error and returns INVALID_ID.

---

int **screen_get_width**(screen: int = 0) const 🔗

Returns the window width matching the graphics API context for the given window ID (in pixels). Despite the parameter being named `screen`, this returns the *window* size. See also screen_get_height.

Note: Only the main **RenderingDevice** returned by RenderingServer.get_rendering_device has a width. If called on a local **RenderingDevice**, this method prints an error and returns INVALID_ID.

---

void **set_resource_name**(id: RID, name: String) 🔗

Sets the resource name for `id` to `name`. This is used for debugging with third-party tools such as RenderDoc [https://renderdoc.org/].

The following types of resources can be named: texture, sampler, vertex buffer, index buffer, uniform buffer, texture buffer, storage buffer, uniform set buffer, shader, render pipeline and compute pipeline. Framebuffers cannot be named. Attempting to name an incompatible resource type will print an error.

Note: Resource names are only set when the engine runs in verbose mode (OS.is_stdout_verbose = `true`), or when using an engine build compiled with the `dev_mode=yes` SCons option. The graphics driver must also support the `VK_EXT_DEBUG_UTILS_EXTENSION_NAME` Vulkan extension for named resources to work.

PackedByteArray **shader_compile_binary_from_spirv**(spirv_data: RDShaderSPIRV, name: String = "") 🔗

Compiles a binary shader from `spirv_data` and returns the compiled binary data as a PackedByteArray. This compiled shader is specific to the GPU model and driver version used; it will not work on different

GPU models or even different driver versions. See also shader_compile_spirv_from_source.

`name` is an optional human-readable name that can be given to the compiled shader for organizational purposes.

RDShaderSPIRV **shader_compile_spirv_from_source**(shader_source: RDShaderSource, allow_cache: bool = true) 🔗

Compiles a SPIR-V from the shader source code in `shader_source` and returns the SPIR-V as a RDShaderSPIRV. This intermediate language shader is portable across different GPU models and driver versions, but cannot be run directly by GPUs until compiled into a binary shader using shader_compile_binary_from_spirv.

If `allow_cache` is `true`, make use of the shader cache generated by Godot. This avoids a potentially lengthy shader compilation step if the shader is already in cache. If `allow_cache` is `false`, Godot's shader cache is ignored and the shader will always be recompiled.

RID **shader_create_from_bytecode**(binary_data: PackedByteArray, placeholder_rid: RID = RID()) 🔗

Creates a new shader instance from a binary compiled shader. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method. See also shader_compile_binary_from_spirv and shader_create_from_spirv.

RID **shader_create_from_spirv**(spirv_data: RDShaderSPIRV, name: String = "") 🔗

Creates a new shader instance from SPIR-V intermediate code. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method. See also shader_compile_spirv_from_source and shader_create_from_bytecode.

RID **shader_create_placeholder**() 🔗

Create a placeholder RID by allocating an RID without initializing it for use in shader_create_from_bytecode. This allows you to create an RID for a shader and pass it around, but defer compiling the shader to a later time.

---

int **shader_get_vertex_input_attribute_mask**(shader: RID) 🔗

Returns the internal vertex input mask. Internally, the vertex input mask is an unsigned integer consisting of the locations (specified in GLSL via. `layout(location = ...)`) of the input variables (specified in GLSL by the `in` keyword).

RID **storage_buffer_create**(size_bytes: int, data: PackedByteArray = PackedByteArray(), usage: BitField[StorageBufferUsage] = 0) 🔗

Creates a storage buffer [https://vkguide.dev/docs/chapter-4/storage_buffers/] with the specified `data` and `usage`. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

---

void **submit**() 🔗

Pushes the frame setup and draw command buffers then marks the local device as currently processing (which allows calling sync).

Note: Only available in local RenderingDevices.

---

void **sync**() 🔗

Forces a synchronization between the CPU and GPU, which may be required in certain cases. Only call this when needed, as CPU-GPU synchronization has a performance cost.

Note: Only available in local RenderingDevices.

Note: sync can only be called after a submit.

RID **texture_buffer_create**(size_bytes: int, format: DataFormat, data: PackedByteArray = PackedByteArray()) 🔗

Creates a new texture buffer. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

Error **texture_clear**(texture: RID, color: Color, base_mipmap: int, mipmap_count: int, base_layer: int, layer_count: int) 🔗

Clears the specified `texture` by replacing all of its pixels with the specified `color`. `base_mipmap` and `mipmap_count` determine which mipmaps of the texture are affected by this clear operation, while `base_layer` and `layer_count` determine which layers of a 3D texture (or texture array) are affected by this clear operation. For 2D textures (which only have one layer by design), `base_layer` must be `0` and `layer_count` must be `1`.

Note: `texture` can't be cleared while a draw list that uses it as part of a framebuffer is being created. Ensure the draw list is finalized (and that the color/depth texture using it is not set to FINAL_ACTION_CONTINUE) to clear this texture.

Error **texture_copy**(from_texture: RID, to_texture: RID, from_pos: Vector3, to_pos: Vector3, size: Vector3, src_mipmap: int, dst_mipmap: int, src_layer: int, dst_layer: int) 🔗

Copies the `from_texture` to `to_texture` with the specified `from_pos`, `to_pos` and `size` coordinates. The Z axis of the `from_pos`, `to_pos` and `size` must be `0` for 2-dimensional textures. Source and destination mipmaps/layers must also be specified, with these parameters being `0` for textures without mipmaps or single-layer textures. Returns @GlobalScope.OK if the texture copy was successful or @GlobalScope.ERR_INVALID_PARAMETER otherwise.

Note: `from_texture` texture can't be copied while a draw list that uses it as part of a framebuffer is being created. Ensure the draw list is finalized (and that the color/depth texture using it is not set to FINAL_ACTION_CONTINUE) to copy this texture.

Note: `from_texture` texture requires the TEXTURE_USAGE_CAN_COPY_FROM_BIT to be retrieved.

Note: `to_texture` can't be copied while a draw list that uses it as part of a framebuffer is being created. Ensure the draw list is finalized (and that the color/depth texture using it is not set to FINAL_ACTION_CONTINUE) to copy this texture.

Note: `to_texture` requires the TEXTURE_USAGE_CAN_COPY_TO_BIT to be retrieved.

Note: `from_texture` and `to_texture` must be of the same type (color or depth).

RID **texture_create**(format: RDTextureFormat, view: RDTextureView, data: Array[PackedByteArray] = []) 🔗

Creates a new texture. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

Note: Not to be confused with RenderingServer.texture_2d_create, which creates the Godot-specific Texture2D resource as opposed to the graphics API's own texture type.

RID **texture_create_from_extension**(type: TextureType, format: DataFormat, samples: TextureSamples, usage_flags: BitField[TextureUsageBits], image: int, width: int, height: int, depth: int, layers: int) 🔗

Returns an RID for an existing `image` (`VkImage`) with the given `type`, `format`, `samples`, `usage_flags`, `width`, `height`, `depth`, and `layers`. This can be used to allow Godot to render onto foreign images.

RID **texture_create_shared**(view: RDTextureView, with_texture: RID) 🔗

Creates a shared texture using the specified `view` and the texture information from `with_texture`.

RID **texture_create_shared_from_slice**(view: RDTextureView, with_texture: RID, layer: int, mipmap: int, mipmaps: int = 1, slice_type: TextureSliceType = 0) 🔗

Creates a shared texture using the specified `view` and the texture information from `with_texture`'s `layer` and `mipmap`. The number of included mipmaps from the original texture can be controlled using the `mipmaps` parameter. Only relevant for textures with multiple layers, such as 3D textures, texture arrays and cubemaps. For single-layer textures, use texture_create_shared

For 2D textures (which only have one layer), `layer` must be `0`.

Note: Layer slicing is only supported for 2D texture arrays, not 3D textures or cubemaps.

PackedByteArray **texture_get_data**(texture: RID, layer: int) 🔗

Returns the `texture` data for the specified `layer` as raw binary data. For 2D textures (which only have one layer), `layer` must be `0`.

Note: `texture` can't be retrieved while a draw list that uses it as part of a framebuffer is being created. Ensure the draw list is finalized (and that the color/depth texture using it is not set to FINAL_ACTION_CONTINUE) to retrieve this texture. Otherwise, an error is printed and a empty PackedByteArray is returned.

Note: `texture` requires the TEXTURE_USAGE_CAN_COPY_FROM_BIT to be retrieved. Otherwise, an error is printed and a empty PackedByteArray is returned.

RDTextureFormat **texture_get_format**(texture: RID) 🔗

Returns the data format used to create this texture.

---

int **texture_get_native_handle**(texture: RID) 🔗

Deprecated: Use get_driver_resource with DRIVER_RESOURCE_TEXTURE instead.

Returns the internal graphics handle for this texture object. For use when communicating with third-party APIs mostly with GDExtension.

Note: This function returns a `uint64_t` which internally maps to a `GLuint` (OpenGL) or `VkImage` (Vulkan).

---

bool **texture_is_format_supported_for_usage**(format: DataFormat, usage_flags: BitField[TextureUsageBits]) const

Returns `true` if the specified `format` is supported for the given `usage_flags`, `false` otherwise.

---

bool **texture_is_shared**(texture: RID)

Returns `true` if the `texture` is shared, `false` otherwise. See RDTextureView.

---

bool **texture_is_valid**(texture: RID)

Returns `true` if the `texture` is valid, `false` otherwise.

Error **texture_resolve_multisample**(from_texture: RID, to_texture: RID)

Resolves the `from_texture` texture onto `to_texture` with multisample antialiasing enabled. This must be used when rendering a framebuffer for MSAA to work. Returns @GlobalScope.OK if successful, @GlobalScope.ERR_INVALID_PARAMETER otherwise.

Note: `from_texture` and `to_texture` textures must have the same dimension, format and type (color or depth).

Note: `from_texture` can't be copied while a draw list that uses it as part of a framebuffer is being created. Ensure the draw list is finalized (and that the color/depth texture using it is not set to FINAL_ACTION_CONTINUE) to resolve this texture.

Note: `from_texture` requires the TEXTURE_USAGE_CAN_COPY_FROM_BIT to be retrieved.

Note: `from_texture` must be multisampled and must also be 2D (or a slice of a 3D/cubemap texture).

Note: `to_texture` can't be copied while a draw list that uses it as part of a framebuffer is being created. Ensure the draw list is finalized (and that the color/depth texture using it is not set to FINAL_ACTION_CONTINUE) to resolve this texture.

Note: `to_texture` texture requires the TEXTURE_USAGE_CAN_COPY_TO_BIT to be retrieved.

Note: `to_texture` texture must **not** be multisampled and must also be 2D (or a slice of a 3D/cubemap texture).

Error **texture_update**(texture: RID, layer: int, data: PackedByteArray)

Updates texture data with new data, replacing the previous data in place. The updated texture data must have the same dimensions and format. For 2D textures (which only have one layer), `layer` must be `0`. Returns @GlobalScope.OK if the update was successful, @GlobalScope.ERR_INVALID_PARAMETER otherwise.

Note: Updating textures is forbidden during creation of a draw or compute list.

Note: The existing `texture` can't be updated while a draw list that uses it as part of a framebuffer is being created. Ensure the draw list is finalized (and that the color/depth texture using it is not set to FINAL_ACTION_CONTINUE) to update this texture.

Note: The existing `texture` requires the TEXTURE_USAGE_CAN_UPDATE_BIT to be updatable.

RID **uniform_buffer_create**(size_bytes: int, data: PackedByteArray = PackedByteArray())

Creates a new uniform buffer. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

RID **uniform_set_create**(uniforms: Array[RDUniform], shader: RID, shader_set: int)

Creates a new uniform set. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

---

bool **uniform_set_is_valid**(uniform_set: RID)

Checks if the `uniform_set` is valid, i.e. is owned.

RID **vertex_array_create**(vertex_count: int, vertex_format: int, src_buffers: Array[RID], offsets: PackedInt64Array = PackedInt64Array())

Creates a vertex array based on the specified buffers. Optionally, `offsets` (in bytes) may be defined for each buffer.

RID **vertex_buffer_create**(size_bytes: int, data: PackedByteArray = PackedByteArray(), use_as_storage: bool = false) 🔗

It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingDevice's free_rid method.

---

int **vertex_format_create**(vertex_descriptions: Array[RDVertexAttribute]) 🔗

Creates a new vertex format with the specified `vertex_descriptions`. Returns a unique vertex format ID corresponding to the newly created vertex format.

RenderingServer

Inherits: Object

Server for anything visible.

Description

The rendering server is the API backend for everything visible. The whole scene system mounts on it to display. The rendering server is completely opaque: the internals are entirely implementation-specific and cannot be accessed.

The rendering server can be used to bypass the scene/Node system entirely. This can improve performance in cases where the scene system is the bottleneck, but won't improve performance otherwise (for instance, if the GPU is already fully utilized).

Resources are created using the `*_create` functions. These functions return RIDs which are not references to the objects themselves, but opaque *pointers* towards these objects.

All objects are drawn to a viewport. You can use the Viewport attached to the SceneTree or you can create one yourself with viewport_create. When using a custom scenario or canvas, the scenario or canvas needs to be attached to the viewport using viewport_set_scenario or viewport_attach_canvas.

Scenarios: In 3D, all visual objects must be associated with a scenario. The scenario is a visual representation of the world. If accessing the rendering server from a running game, the scenario can be accessed from the scene tree from any Node3D node with Node3D.get_world_3d. Otherwise, a scenario can be created with scenario_create.

Similarly, in 2D, a canvas is needed to draw all canvas items.

3D: In 3D, all visible objects are comprised of a resource and an instance. A resource can be a mesh, a particle system, a light, or any other 3D object. In order to be visible resources must be attached to an instance using instance_set_base. The instance must also be attached to the scenario using instance_set_scenario in order to be visible. RenderingServer methods that don't have a prefix are usually 3D-specific (but not always).

2D: In 2D, all visible objects are some form of canvas item. In order to be visible, a canvas item needs to be the child of a canvas attached to a viewport, or it needs to be the child of another canvas item that is eventually attached to the canvas. 2D-specific RenderingServer methods generally start with `canvas_*`.

Headless mode: Starting the engine with the `--headless` command line argument disables all rendering and window management functions. Most functions from **RenderingServer** will return dummy values in this case.

Tutorials

- Optimization using Servers

Properties

bool	render_loop_enabled

Methods

Array[Image]	bake_render_uv2(base: RID, material_overrides: Array[RID], image_size: Vector2i)
void	call_on_render_thread(callable: Callable)
RID	camera_attributes_create()
void	camera_attributes_set_auto_exposure(camera_attributes: RID, enable: bool, min_sensitivity: float, max_sensitivity: float, speed: float, scale: float)
void	camera_attributes_set_dof_blur(camera_attributes: RID, far_enable: bool, far_distance: float, far_transition: float, near_enable: bool, near_distance: float, near_transition: float, amount: float)
void	camera_attributes_set_dof_blur_bokeh_shape(shape: DOFBokehShape)
void	camera_attributes_set_dof_blur_quality(quality: DOFBlurQuality, use_jitter: bool)
void	camera_attributes_set_exposure(camera_attributes: RID, multiplier: float, normalization: float)
RID	camera_create()
void	camera_set_camera_attributes(camera: RID, effects: RID)
void	camera_set_compositor(camera: RID, compositor: RID)
void	camera_set_cull_mask(camera: RID, layers: int)

void	camera_set_environment(camera: RID, env: RID)
void	camera_set_frustum(camera: RID, size: float, offset: Vector2, z_near: float, z_far: float)
void	camera_set_orthogonal(camera: RID, size: float, z_near: float, z_far: float)
void	camera_set_perspective(camera: RID, fovy_degrees: float, z_near: float, z_far: float)
void	camera_set_transform(camera: RID, transform: Transform3D)
void	camera_set_use_vertical_aspect(camera: RID, enable: bool)
RID	canvas_create()
void	canvas_item_add_animation_slice(item: RID, animation_length: float, slice_begin: float, slice_end: float, offset: float = 0.0)
void	canvas_item_add_circle(item: RID, pos: Vector2, radius: float, color: Color, antialiased: bool = false)
void	canvas_item_add_clip_ignore(item: RID, ignore: bool)
void	canvas_item_add_lcd_texture_rect_region(item: RID, rect: Rect2, texture: RID, src_rect: Rect2, modulate: Color)
void	canvas_item_add_line(item: RID, from: Vector2, to: Vector2, color: Color, width: float = -1.0, antialiased: bool = false)
void	canvas_item_add_mesh(item: RID, mesh: RID, transform: Transform2D = Transform2D(1, 0, 0, 1, 0, 0), modulate: Color = Color(1, 1, 1, 1), texture: RID = RID())
void	canvas_item_add_msdf_texture_rect_region(item: RID, rect: Rect2, texture: RID, src_rect: Rect2, modulate: Color = Color(1, 1, 1, 1), outline_size: int = 0, px_range: float = 1.0, scale: float = 1.0)

void	canvas_item_add_multiline(item: RID, points: PackedVector2Array, colors: PackedColorArray, width: float = -1.0, antialiased: bool = false)
void	canvas_item_add_multimesh(item: RID, mesh: RID, texture: RID = RID())
void	canvas_item_add_nine_patch(item: RID, rect: Rect2, source: Rect2, texture: RID, topleft: Vector2, bottomright: Vector2, x_axis_mode: NinePatchAxisMode = 0, y_axis_mode: NinePatchAxisMode = 0, draw_center: bool = true, modulate: Color = Color(1, 1, 1, 1))
void	canvas_item_add_particles(item: RID, particles: RID, texture: RID)
void	canvas_item_add_polygon(item: RID, points: PackedVector2Array, colors: PackedColorArray, uvs: PackedVector2Array = PackedVector2Array(), texture: RID = RID())
void	canvas_item_add_polyline(item: RID, points: PackedVector2Array, colors: PackedColorArray, width: float = -1.0, antialiased: bool = false)
void	canvas_item_add_primitive(item: RID, points: PackedVector2Array, colors: PackedColorArray, uvs: PackedVector2Array, texture: RID)
void	canvas_item_add_rect(item: RID, rect: Rect2, color: Color, antialiased: bool = false)
void	canvas_item_add_set_transform(item: RID, transform: Transform2D)
void	canvas_item_add_texture_rect(item: RID, rect: Rect2, texture: RID, tile: bool = false, modulate: Color = Color(1, 1, 1, 1), transpose: bool = false)
void	canvas_item_add_texture_rect_region(item: RID, rect: Rect2, texture: RID, src_rect: Rect2, modulate: Color = Color(1, 1, 1, 1), transpose: bool = false, clip_uv: bool = true)
void	canvas_item_add_triangle_array(item: RID, indices: PackedInt32Array, points: PackedVector2Array, colors: PackedColorArray, uvs: PackedVector2Array = PackedVector2Array(), bones: PackedInt32Array = PackedInt32Array(), weights: PackedFloat32Array = PackedFloat32Array(), texture: RID = RID(), count: int = -1)

void	canvas_item_clear(item: RID)
RID	canvas_item_create()
void	canvas_item_reset_physics_interpolation(item: RID)
void	canvas_item_set_canvas_group_mode(item: RID, mode: CanvasGroupMode, clear_margin: float = 5.0, fit_empty: bool = false, fit_margin: float = 0.0, blur_mipmaps: bool = false)
void	canvas_item_set_clip(item: RID, clip: bool)
void	canvas_item_set_copy_to_backbuffer(item: RID, enabled: bool, rect: Rect2)
void	canvas_item_set_custom_rect(item: RID, use_custom_rect: bool, rect: Rect2 = Rect2(0, 0, 0, 0))
void	canvas_item_set_default_texture_filter(item: RID, filter: CanvasItemTextureFilter)
void	canvas_item_set_default_texture_repeat(item: RID, repeat: CanvasItemTextureRepeat)
void	canvas_item_set_distance_field_mode(item: RID, enabled: bool)
void	canvas_item_set_draw_behind_parent(item: RID, enabled: bool)
void	canvas_item_set_draw_index(item: RID, index: int)
void	canvas_item_set_interpolated(item: RID, interpolated: bool)
void	canvas_item_set_light_mask(item: RID, mask: int)
void	canvas_item_set_material(item: RID, material: RID)
void	canvas_item_set_modulate(item: RID, color: Color)

void	canvas_item_set_parent(item: RID, parent: RID)
void	canvas_item_set_self_modulate(item: RID, color: Color)
void	canvas_item_set_sort_children_by_y(item: RID, enabled: bool)
void	canvas_item_set_transform(item: RID, transform: Transform2D)
void	canvas_item_set_use_parent_material(item: RID, enabled: bool)
void	canvas_item_set_visibility_layer(item: RID, visibility_layer: int)
void	canvas_item_set_visibility_notifier(item: RID, enable: bool, area: Rect2, enter_callable: Callable, exit_callable: Callable)
void	canvas_item_set_visible(item: RID, visible: bool)
void	canvas_item_set_z_as_relative_to_parent(item: RID, enabled: bool)
void	canvas_item_set_z_index(item: RID, z_index: int)
void	canvas_item_transform_physics_interpolation(item: RID, transform: Transform2D)
void	canvas_light_attach_to_canvas(light: RID, canvas: RID)
RID	canvas_light_create()
void	canvas_light_occluder_attach_to_canvas(occluder: RID, canvas: RID)
RID	canvas_light_occluder_create()
void	canvas_light_occluder_reset_physics_interpolation(occluder: RID)
void	canvas_light_occluder_set_as_sdf_collision(occluder: RID, enable: bool)

void	canvas_light_occluder_set_enabled(occluder: RID, enabled: bool)
void	canvas_light_occluder_set_interpolated(occluder: RID, interpolated: bool)
void	canvas_light_occluder_set_light_mask(occluder: RID, mask: int)
void	canvas_light_occluder_set_polygon(occluder: RID, polygon: RID)
void	canvas_light_occluder_set_transform(occluder: RID, transform: Transform2D)
void	canvas_light_occluder_transform_physics_interpolation(occluder: RID, transform: Transform2D)
void	canvas_light_reset_physics_interpolation(light: RID)
void	canvas_light_set_blend_mode(light: RID, mode: CanvasLightBlendMode)
void	canvas_light_set_color(light: RID, color: Color)
void	canvas_light_set_enabled(light: RID, enabled: bool)
void	canvas_light_set_energy(light: RID, energy: float)
void	canvas_light_set_height(light: RID, height: float)
void	canvas_light_set_interpolated(light: RID, interpolated: bool)
void	canvas_light_set_item_cull_mask(light: RID, mask: int)
void	canvas_light_set_item_shadow_cull_mask(light: RID, mask: int)
void	canvas_light_set_layer_range(light: RID, min_layer: int, max_layer: int)
void	canvas_light_set_mode(light: RID, mode: CanvasLightMode)

void	canvas_light_set_shadow_color(light: RID, color: Color)
void	canvas_light_set_shadow_enabled(light: RID, enabled: bool)
void	canvas_light_set_shadow_filter(light: RID, filter: CanvasLightShadowFilter)
void	canvas_light_set_shadow_smooth(light: RID, smooth: float)
void	canvas_light_set_texture(light: RID, texture: RID)
void	canvas_light_set_texture_offset(light: RID, offset: Vector2)
void	canvas_light_set_texture_scale(light: RID, scale: float)
void	canvas_light_set_transform(light: RID, transform: Transform2D)
void	canvas_light_set_z_range(light: RID, min_z: int, max_z: int)
void	canvas_light_transform_physics_interpolation(light: RID, transform: Transform2D)
RID	canvas_occluder_polygon_create()
void	canvas_occluder_polygon_set_cull_mode(occluder_polygon: RID, mode: CanvasOccluderPolygonCullMode)
void	canvas_occluder_polygon_set_shape(occluder_polygon: RID, shape: PackedVector2Array, closed: bool)
void	canvas_set_disable_scale(disable: bool)
void	canvas_set_item_mirroring(canvas: RID, item: RID, mirroring: Vector2)
void	canvas_set_item_repeat(item: RID, repeat_size: Vector2, repeat_times: int)

void	canvas_set_modulate(canvas: RID, color: Color)
void	canvas_set_shadow_texture_size(size: int)
RID	canvas_texture_create()
void	canvas_texture_set_channel(canvas_texture: RID, channel: CanvasTextureChannel, texture: RID)
void	canvas_texture_set_shading_parameters(canvas_texture: RID, base_color: Color, shininess: float)
void	canvas_texture_set_texture_filter(canvas_texture: RID, filter: CanvasItemTextureFilter)
void	canvas_texture_set_texture_repeat(canvas_texture: RID, repeat: CanvasItemTextureRepeat)
RID	compositor_create()
RID	compositor_effect_create()
void	compositor_effect_set_callback(effect: RID, callback_type: CompositorEffectCallbackType, callback: Callable)
void	compositor_effect_set_enabled(effect: RID, enabled: bool)
void	compositor_effect_set_flag(effect: RID, flag: CompositorEffectFlags, set: bool)
void	compositor_set_compositor_effects(compositor: RID, effects: Array[RID])
RenderingDevice	create_local_rendering_device() const
Rect2	debug_canvas_item_get_rect(item: RID)
RID	decal_create()

void	decal_set_albedo_mix(decal: RID, albedo_mix: float)
void	decal_set_cull_mask(decal: RID, mask: int)
void	decal_set_distance_fade(decal: RID, enabled: bool, begin: float, length: float)
void	decal_set_emission_energy(decal: RID, energy: float)
void	decal_set_fade(decal: RID, above: float, below: float)
void	decal_set_modulate(decal: RID, color: Color)
void	decal_set_normal_fade(decal: RID, fade: float)
void	decal_set_size(decal: RID, size: Vector3)
void	decal_set_texture(decal: RID, type: DecalTexture, texture: RID)
void	decals_set_filter(filter: DecalFilter)
RID	directional_light_create()
void	directional_shadow_atlas_set_size(size: int, is_16bits: bool)
void	directional_soft_shadow_filter_set_quality(quality: ShadowQuality)
Image	environment_bake_panorama(environment: RID, bake_irradiance: bool, size: Vector2i)
RID	environment_create()
void	environment_glow_set_use_bicubic_upscale(enable: bool)
void	environment_set_adjustment(env: RID, enable: bool, brightness: float, contrast: float, saturation: float, use_1d_color_correction: bool, color_correction: RID)

void	environment_set_ambient_light(env: RID, color: Color, ambient: EnvironmentAmbientSource = 0, energy: float = 1.0, sky_contibution: float = 0.0, reflection_source: EnvironmentReflectionSource = 0)
void	environment_set_background(env: RID, bg: EnvironmentBG)
void	environment_set_bg_color(env: RID, color: Color)
void	environment_set_bg_energy(env: RID, multiplier: float, exposure_value: float)
void	environment_set_canvas_max_layer(env: RID, max_layer: int)
void	environment_set_fog(env: RID, enable: bool, light_color: Color, light_energy: float, sun_scatter: float, density: float, height: float, height_density: float, aerial_perspective: float, sky_affect: float, fog_mode: EnvironmentFogMode = 0)
void	environment_set_glow(env: RID, enable: bool, levels: PackedFloat32Array, intensity: float, strength: float, mix: float, bloom_threshold: float, blend_mode: EnvironmentGlowBlendMode, hdr_bleed_threshold: float, hdr_bleed_scale: float, hdr_luminance_cap: float, glow_map_strength: float, glow_map: RID)
void	environment_set_sdfgi(env: RID, enable: bool, cascades: int, min_cell_size: float, y_scale: EnvironmentSDFGIYScale, use_occlusion: bool, bounce_feedback: float, read_sky: bool, energy: float, normal_bias: float, probe_bias: float)
void	environment_set_sdfgi_frames_to_converge(frames: EnvironmentSDFGIFramesToConverge)
void	environment_set_sdfgi_frames_to_update_light(frames: EnvironmentSDFGIFramesToUpdateLight)
void	environment_set_sdfgi_ray_count(ray_count: EnvironmentSDFGIRayCount)
void	environment_set_sky(env: RID, sky: RID)
void	environment_set_sky_custom_fov(env: RID, scale: float)

void	environment_set_sky_orientation(env: RID, orientation: Basis)
void	environment_set_ssao(env: RID, enable: bool, radius: float, intensity: float, power: float, detail: float, horizon: float, sharpness: float, light_affect: float, ao_channel_affect: float)
void	environment_set_ssao_quality(quality: EnvironmentSSAOQuality, half_size: bool, adaptive_target: float, blur_passes: int, fadeout_from: float, fadeout_to: float)
void	environment_set_ssil_quality(quality: EnvironmentSSILQuality, half_size: bool, adaptive_target: float, blur_passes: int, fadeout_from: float, fadeout_to: float)
void	environment_set_ssr(env: RID, enable: bool, max_steps: int, fade_in: float, fade_out: float, depth_tolerance: float)
void	environment_set_ssr_roughness_quality(quality: EnvironmentSSRRoughnessQuality)
void	environment_set_tonemap(env: RID, tone_mapper: EnvironmentToneMapper, exposure: float, white: float)
void	environment_set_volumetric_fog(env: RID, enable: bool, density: float, albedo: Color, emission: Color, emission_energy: float, anisotropy: float, length: float, p_detail_spread: float, gi_inject: float, temporal_reprojection: bool, temporal_reprojection_amount: float, ambient_inject: float, sky_affect: float)
void	environment_set_volumetric_fog_filter_active(active: bool)
void	environment_set_volumetric_fog_volume_size(size: int, depth: int)
RID	fog_volume_create()
void	fog_volume_set_material(fog_volume: RID, material: RID)
void	fog_volume_set_shape(fog_volume: RID, shape: FogVolumeShape)
void	fog_volume_set_size(fog_volume: RID, size: Vector3)

void	force_draw(swap_buffers: bool = true, frame_step: float = 0.0)
void	force_sync()
void	free_rid(rid: RID)
Color	get_default_clear_color()
float	get_frame_setup_time_cpu() const
RenderingDevice	get_rendering_device() const
int	get_rendering_info(info: RenderingInfo)
Array[Dictionary]	get_shader_parameter_list(shader: RID) const
RID	get_test_cube()
RID	get_test_texture()
String	get_video_adapter_api_version() const
String	get_video_adapter_name() const
DeviceType	get_video_adapter_type() const
String	get_video_adapter_vendor() const
RID	get_white_texture()
void	gi_set_use_half_resolution(half_resolution: bool)
void	global_shader_parameter_add(name: StringName, type: GlobalShaderParameterType, default_value: Variant)
Variant	global_shader_parameter_get(name: StringName) const

Array[StringName]	global_shader_parameter_get_list() const
GlobalShaderParameterType	global_shader_parameter_get_type(name: StringName) const
void	global_shader_parameter_remove(name: StringName)
void	global_shader_parameter_set(name: StringName, value: Variant)
void	global_shader_parameter_set_override(name: StringName, value: Variant)
bool	has_changed() const
bool	has_feature(feature: Features) const
bool	has_os_feature(feature: String) const
void	instance_attach_object_instance_id(instance: RID, id: int)
void	instance_attach_skeleton(instance: RID, skeleton: RID)
RID	instance_create()
RID	instance_create2(base: RID, scenario: RID)
Variant	instance_geometry_get_shader_parameter(instance: RID, parameter: StringName) const
Variant	instance_geometry_get_shader_parameter_default_value(instance: RID, parameter: StringName) const
Array[Dictionary]	instance_geometry_get_shader_parameter_list(instance: RID) const
void	instance_geometry_set_cast_shadows_setting(instance: RID, shadow_casting_setting: ShadowCastingSetting)
void	instance_geometry_set_flag(instance: RID, flag: InstanceFlags, enabled: bool)

void	instance_geometry_set_lightmap(instance: RID, lightmap: RID, lightmap_uv_scale: Rect2, lightmap_slice: int)
void	instance_geometry_set_lod_bias(instance: RID, lod_bias: float)
void	instance_geometry_set_material_overlay(instance: RID, material: RID)
void	instance_geometry_set_material_override(instance: RID, material: RID)
void	instance_geometry_set_shader_parameter(instance: RID, parameter: StringName, value: Variant)
void	instance_geometry_set_transparency(instance: RID, transparency: float)
void	instance_geometry_set_visibility_range(instance: RID, min: float, max: float, min_margin: float, max_margin: float, fade_mode: VisibilityRangeFadeMode)
void	instance_set_base(instance: RID, base: RID)
void	instance_set_blend_shape_weight(instance: RID, shape: int, weight: float)
void	instance_set_custom_aabb(instance: RID, aabb: AABB)
void	instance_set_extra_visibility_margin(instance: RID, margin: float)
void	instance_set_ignore_culling(instance: RID, enabled: bool)
void	instance_set_layer_mask(instance: RID, mask: int)
void	instance_set_pivot_data(instance: RID, sorting_offset: float, use_aabb_center: bool)
void	instance_set_scenario(instance: RID, scenario: RID)

void	instance_set_surface_override_material(instance: RID, surface: int, material: RID)
void	instance_set_transform(instance: RID, transform: Transform3D)
void	instance_set_visibility_parent(instance: RID, parent: RID)
void	instance_set_visible(instance: RID, visible: bool)
PackedInt64Array	instances_cull_aabb(aabb: AABB, scenario: RID = RID()) const
PackedInt64Array	instances_cull_convex(convex: Array[Plane], scenario: RID = RID()) const
PackedInt64Array	instances_cull_ray(from: Vector3, to: Vector3, scenario: RID = RID()) const
bool	is_on_render_thread()
void	light_directional_set_blend_splits(light: RID, enable: bool)
void	light_directional_set_shadow_mode(light: RID, mode: LightDirectionalShadowMode)
void	light_directional_set_sky_mode(light: RID, mode: LightDirectionalSkyMode)
void	light_omni_set_shadow_mode(light: RID, mode: LightOmniShadowMode)
void	light_projectors_set_filter(filter: LightProjectorFilter)
void	light_set_bake_mode(light: RID, bake_mode: LightBakeMode)
void	light_set_color(light: RID, color: Color)
void	light_set_cull_mask(light: RID, mask: int)

void	light_set_distance_fade(decal: RID, enabled: bool, begin: float, shadow: float, length: float)
void	light_set_max_sdfgi_cascade(light: RID, cascade: int)
void	light_set_negative(light: RID, enable: bool)
void	light_set_param(light: RID, param: LightParam, value: float)
void	light_set_projector(light: RID, texture: RID)
void	light_set_reverse_cull_face_mode(light: RID, enabled: bool)
void	light_set_shadow(light: RID, enabled: bool)
RID	lightmap_create()
PackedInt32Array	lightmap_get_probe_capture_bsp_tree(lightmap: RID) const
PackedVector3Array	lightmap_get_probe_capture_points(lightmap: RID) const
PackedColorArray	lightmap_get_probe_capture_sh(lightmap: RID) const
PackedInt32Array	lightmap_get_probe_capture_tetrahedra(lightmap: RID) const
void	lightmap_set_baked_exposure_normalization(lightmap: RID, baked_exposure: float)
void	lightmap_set_probe_bounds(lightmap: RID, bounds: AABB)
void	lightmap_set_probe_capture_data(lightmap: RID, points: PackedVector3Array, point_sh: PackedColorArray, tetrahedra: PackedInt32Array, bsp_tree: PackedInt32Array)
void	lightmap_set_probe_capture_update_speed(speed: float)
void	lightmap_set_probe_interior(lightmap: RID, interior: bool)

void	lightmap_set_textures(lightmap: RID, light: RID, uses_sh: bool)
RID	make_sphere_mesh(latitudes: int, longitudes: int, radius: float)
RID	material_create()
Variant	material_get_param(material: RID, parameter: StringName) const
void	material_set_next_pass(material: RID, next_material: RID)
void	material_set_param(material: RID, parameter: StringName, value: Variant)
void	material_set_render_priority(material: RID, priority: int)
void	material_set_shader(shader_material: RID, shader: RID)
void	mesh_add_surface(mesh: RID, surface: Dictionary)
void	mesh_add_surface_from_arrays(mesh: RID, primitive: PrimitiveType, arrays: Array, blend_shapes: Array = [], lods: Dictionary = {}, compress_format: BitField[ArrayFormat] = 0)
void	mesh_clear(mesh: RID)
RID	mesh_create()
RID	mesh_create_from_surfaces(surfaces: Array[Dictionary], blend_shape_count: int = 0)
int	mesh_get_blend_shape_count(mesh: RID) const
BlendShapeMode	mesh_get_blend_shape_mode(mesh: RID) const
AABB	mesh_get_custom_aabb(mesh: RID) const
Dictionary	mesh_get_surface(mesh: RID, surface: int)

int	mesh_get_surface_count(mesh: RID) const
void	mesh_set_blend_shape_mode(mesh: RID, mode: BlendShapeMode)
void	mesh_set_custom_aabb(mesh: RID, aabb: AABB)
void	mesh_set_shadow_mesh(mesh: RID, shadow_mesh: RID)
Array	mesh_surface_get_arrays(mesh: RID, surface: int) const
Array[Array]	mesh_surface_get_blend_shape_arrays(mesh: RID, surface: int) const
int	mesh_surface_get_format_attribute_stride(format: BitField[ArrayFormat], vertex_count: int) const
int	mesh_surface_get_format_normal_tangent_stride(format: BitField[ArrayFormat], vertex_count: int) const
int	mesh_surface_get_format_offset(format: BitField[ArrayFormat], vertex_count: int, array_index: int) const
int	mesh_surface_get_format_skin_stride(format: BitField[ArrayFormat], vertex_count: int) const
int	mesh_surface_get_format_vertex_stride(format: BitField[ArrayFormat], vertex_count: int) const
RID	mesh_surface_get_material(mesh: RID, surface: int) const
void	mesh_surface_set_material(mesh: RID, surface: int, material: RID)
void	mesh_surface_update_attribute_region(mesh: RID, surface: int, offset: int, data: PackedByteArray)
void	mesh_surface_update_skin_region(mesh: RID, surface: int, offset: int, data: PackedByteArray)
void	mesh_surface_update_vertex_region(mesh: RID, surface: int, offset: int, data: PackedByteArray)

void	multimesh_allocate_data(multimesh: RID, instances: int, transform_format: MultimeshTransformFormat, color_format: bool = false, custom_data_format: bool = false)
RID	multimesh_create()
AABB	multimesh_get_aabb(multimesh: RID) const
PackedFloat32Array	multimesh_get_buffer(multimesh: RID) const
AABB	multimesh_get_custom_aabb(multimesh: RID) const
int	multimesh_get_instance_count(multimesh: RID) const
RID	multimesh_get_mesh(multimesh: RID) const
int	multimesh_get_visible_instances(multimesh: RID) const
Color	multimesh_instance_get_color(multimesh: RID, index: int) const
Color	multimesh_instance_get_custom_data(multimesh: RID, index: int) const
Transform3D	multimesh_instance_get_transform(multimesh: RID, index: int) const
Transform2D	multimesh_instance_get_transform_2d(multimesh: RID, index: int) const
void	multimesh_instance_set_color(multimesh: RID, index: int, color: Color)
void	multimesh_instance_set_custom_data(multimesh: RID, index: int, custom_data: Color)
void	multimesh_instance_set_transform(multimesh: RID, index: int, transform: Transform3D)
void	multimesh_instance_set_transform_2d(multimesh: RID, index: int, transform: Transform2D)

void	multimesh_set_buffer(multimesh: RID, buffer: PackedFloat32Array)
void	multimesh_set_custom_aabb(multimesh: RID, aabb: AABB)
void	multimesh_set_mesh(multimesh: RID, mesh: RID)
void	multimesh_set_visible_instances(multimesh: RID, visible: int)
RID	occluder_create()
void	occluder_set_mesh(occluder: RID, vertices: PackedVector3Array, indices: PackedInt32Array)
RID	omni_light_create()
RID	particles_collision_create()
void	particles_collision_height_field_update(particles_collision: RID)
void	particles_collision_set_attractor_attenuation(particles_collision: RID, curve: float)
void	particles_collision_set_attractor_directionality(particles_collision: RID, amount: float)
void	particles_collision_set_attractor_strength(particles_collision: RID, strength: float)
void	particles_collision_set_box_extents(particles_collision: RID, extents: Vector3)
void	particles_collision_set_collision_type(particles_collision: RID, type: ParticlesCollisionType)
void	particles_collision_set_cull_mask(particles_collision: RID, mask: int)
void	particles_collision_set_field_texture(particles_collision: RID, texture: RID)

void	particles_collision_set_height_field_resolution(particles_collision: RID, resolution: ParticlesCollisionHeightfieldResolution)
void	particles_collision_set_sphere_radius(particles_collision: RID, radius: float)
RID	particles_create()
void	particles_emit(particles: RID, transform: Transform3D, velocity: Vector3, color: Color, custom: Color, emit_flags: int)
AABB	particles_get_current_aabb(particles: RID)
bool	particles_get_emitting(particles: RID)
bool	particles_is_inactive(particles: RID)
void	particles_request_process(particles: RID)
void	particles_restart(particles: RID)
void	particles_set_amount(particles: RID, amount: int)
void	particles_set_amount_ratio(particles: RID, ratio: float)
void	particles_set_collision_base_size(particles: RID, size: float)
void	particles_set_custom_aabb(particles: RID, aabb: AABB)
void	particles_set_draw_order(particles: RID, order: ParticlesDrawOrder)
void	particles_set_draw_pass_mesh(particles: RID, pass: int, mesh: RID)
void	particles_set_draw_passes(particles: RID, count: int)
void	particles_set_emission_transform(particles: RID, transform: Transform3D)

void	particles_set_emitter_velocity(particles: RID, velocity: Vector3)
void	particles_set_emitting(particles: RID, emitting: bool)
void	particles_set_explosiveness_ratio(particles: RID, ratio: float)
void	particles_set_fixed_fps(particles: RID, fps: int)
void	particles_set_fractional_delta(particles: RID, enable: bool)
void	particles_set_interp_to_end(particles: RID, factor: float)
void	particles_set_interpolate(particles: RID, enable: bool)
void	particles_set_lifetime(particles: RID, lifetime: float)
void	particles_set_mode(particles: RID, mode: ParticlesMode)
void	particles_set_one_shot(particles: RID, one_shot: bool)
void	particles_set_pre_process_time(particles: RID, time: float)
void	particles_set_process_material(particles: RID, material: RID)
void	particles_set_randomness_ratio(particles: RID, ratio: float)
void	particles_set_speed_scale(particles: RID, scale: float)
void	particles_set_subemitter(particles: RID, subemitter_particles: RID)
void	particles_set_trail_bind_poses(particles: RID, bind_poses: Array[Transform3D])
void	particles_set_trails(particles: RID, enable: bool, length_sec: float)
void	particles_set_transform_align(particles: RID, align: ParticlesTransformAlign)

void	particles_set_use_local_coordinates(particles: RID, enable: bool)
void	positional_soft_shadow_filter_set_quality(quality: ShadowQuality)
RID	reflection_probe_create()
void	reflection_probe_set_ambient_color(probe: RID, color: Color)
void	reflection_probe_set_ambient_energy(probe: RID, energy: float)
void	reflection_probe_set_ambient_mode(probe: RID, mode: ReflectionProbeAmbientMode)
void	reflection_probe_set_as_interior(probe: RID, enable: bool)
void	reflection_probe_set_cull_mask(probe: RID, layers: int)
void	reflection_probe_set_enable_box_projection(probe: RID, enable: bool)
void	reflection_probe_set_enable_shadows(probe: RID, enable: bool)
void	reflection_probe_set_intensity(probe: RID, intensity: float)
void	reflection_probe_set_max_distance(probe: RID, distance: float)
void	reflection_probe_set_mesh_lod_threshold(probe: RID, pixels: float)
void	reflection_probe_set_origin_offset(probe: RID, offset: Vector3)
void	reflection_probe_set_reflection_mask(probe: RID, layers: int)
void	reflection_probe_set_resolution(probe: RID, resolution: int)
void	reflection_probe_set_size(probe: RID, size: Vector3)
void	reflection_probe_set_update_mode(probe: RID, mode: ReflectionProbeUpdateMode)

void	request_frame_drawn_callback(callable: Callable)
RID	scenario_create()
void	scenario_set_camera_attributes(scenario: RID, effects: RID)
void	scenario_set_compositor(scenario: RID, compositor: RID)
void	scenario_set_environment(scenario: RID, environment: RID)
void	scenario_set_fallback_environment(scenario: RID, environment: RID)
void	screen_space_roughness_limiter_set_active(enable: bool, amount: float, limit: float)
void	set_boot_image(image: Image, color: Color, scale: bool, use_filter: bool = true)
void	set_debug_generate_wireframes(generate: bool)
void	set_default_clear_color(color: Color)
RID	shader_create()
String	shader_get_code(shader: RID) const
RID	shader_get_default_texture_parameter(shader: RID, name: StringName, index: int = 0) const
Variant	shader_get_parameter_default(shader: RID, name: StringName) const
void	shader_set_code(shader: RID, code: String)
void	shader_set_default_texture_parameter(shader: RID, name: StringName, texture: RID, index: int = 0)
void	shader_set_path_hint(shader: RID, path: String)

void	skeleton_allocate_data(skeleton: RID, bones: int, is_2d_skeleton: bool = false)
Transform3D	skeleton_bone_get_transform(skeleton: RID, bone: int) const
Transform2D	skeleton_bone_get_transform_2d(skeleton: RID, bone: int) const
void	skeleton_bone_set_transform(skeleton: RID, bone: int, transform: Transform3D)
void	skeleton_bone_set_transform_2d(skeleton: RID, bone: int, transform: Transform2D)
RID	skeleton_create()
int	skeleton_get_bone_count(skeleton: RID) const
void	skeleton_set_base_transform_2d(skeleton: RID, base_transform: Transform2D)
Image	sky_bake_panorama(sky: RID, energy: float, bake_irradiance: bool, size: Vector2i)
RID	sky_create()
void	sky_set_material(sky: RID, material: RID)
void	sky_set_mode(sky: RID, mode: SkyMode)
void	sky_set_radiance_size(sky: RID, radiance_size: int)
RID	spot_light_create()
void	sub_surface_scattering_set_quality(quality: SubSurfaceScatteringQuality)
void	sub_surface_scattering_set_scale(scale: float, depth_scale: float)

RID	texture_2d_create(image: Image)
Image	texture_2d_get(texture: RID) const
Image	texture_2d_layer_get(texture: RID, layer: int) const
RID	texture_2d_layered_create(layers: Array[Image], layered_type: TextureLayeredType)
RID	texture_2d_layered_placeholder_create(layered_type: TextureLayeredType)
RID	texture_2d_placeholder_create()
void	texture_2d_update(texture: RID, image: Image, layer: int)
RID	texture_3d_create(format: Format, width: int, height: int, depth: int, mipmaps: bool, data: Array[Image])
Array[Image]	texture_3d_get(texture: RID) const
RID	texture_3d_placeholder_create()
void	texture_3d_update(texture: RID, data: Array[Image])
Format	texture_get_format(texture: RID) const
int	texture_get_native_handle(texture: RID, srgb: bool = false) const
String	texture_get_path(texture: RID) const
RID	texture_get_rd_texture(texture: RID, srgb: bool = false) const
RID	texture_proxy_create(base: RID)
void	texture_proxy_update(texture: RID, proxy_to: RID)

RID	texture_rd_create(rd_texture: RID, layer_type: TextureLayeredType = 0)
void	texture_replace(texture: RID, by_texture: RID)
void	texture_set_force_redraw_if_visible(texture: RID, enable: bool)
void	texture_set_path(texture: RID, path: String)
void	texture_set_size_override(texture: RID, width: int, height: int)
void	viewport_attach_camera(viewport: RID, camera: RID)
void	viewport_attach_canvas(viewport: RID, canvas: RID)
void	viewport_attach_to_screen(viewport: RID, rect: Rect2 = Rect2(0, 0, 0, 0), screen: int = 0)
RID	viewport_create()
float	viewport_get_measured_render_time_cpu(viewport: RID) const
float	viewport_get_measured_render_time_gpu(viewport: RID) const
int	viewport_get_render_info(viewport: RID, type: ViewportRenderInfoType, info: ViewportRenderInfo)
RID	viewport_get_render_target(viewport: RID) const
RID	viewport_get_texture(viewport: RID) const
ViewportUpdateMode	viewport_get_update_mode(viewport: RID) const
void	viewport_remove_canvas(viewport: RID, canvas: RID)
void	viewport_set_active(viewport: RID, active: bool)

void	viewport_set_canvas_cull_mask(viewport: RID, canvas_cull_mask: int)
void	viewport_set_canvas_stacking(viewport: RID, canvas: RID, layer: int, sublayer: int)
void	viewport_set_canvas_transform(viewport: RID, canvas: RID, offset: Transform2D)
void	viewport_set_clear_mode(viewport: RID, clear_mode: ViewportClearMode)
void	viewport_set_debug_draw(viewport: RID, draw: ViewportDebugDraw)
void	viewport_set_default_canvas_item_texture_filter(viewport: RID, filter: CanvasItemTextureFilter)
void	viewport_set_default_canvas_item_texture_repeat(viewport: RID, repeat: CanvasItemTextureRepeat)
void	viewport_set_disable_2d(viewport: RID, disable: bool)
void	viewport_set_disable_3d(viewport: RID, disable: bool)
void	viewport_set_environment_mode(viewport: RID, mode: ViewportEnvironmentMode)
void	viewport_set_fsr_sharpness(viewport: RID, sharpness: float)
void	viewport_set_global_canvas_transform(viewport: RID, transform: Transform2D)
void	viewport_set_measure_render_time(viewport: RID, enable: bool)
void	viewport_set_msaa_2d(viewport: RID, msaa: ViewportMSAA)
void	viewport_set_msaa_3d(viewport: RID, msaa: ViewportMSAA)
void	viewport_set_occlusion_culling_build_quality(quality: ViewportOcclusionCullingBuildQuality)

void	viewport_set_occlusion_rays_per_thread(rays_per_thread: int)
void	viewport_set_parent_viewport(viewport: RID, parent_viewport: RID)
void	viewport_set_positional_shadow_atlas_quadrant_subdivision(viewport: RID, quadrant: int, subdivision: int)
void	viewport_set_positional_shadow_atlas_size(viewport: RID, size: int, use_16_bits: bool = false)
void	viewport_set_render_direct_to_screen(viewport: RID, enabled: bool)
void	viewport_set_scaling_3d_mode(viewport: RID, scaling_3d_mode: ViewportScaling3DMode)
void	viewport_set_scaling_3d_scale(viewport: RID, scale: float)
void	viewport_set_scenario(viewport: RID, scenario: RID)
void	viewport_set_screen_space_aa(viewport: RID, mode: ViewportScreenSpaceAA)
void	viewport_set_sdf_oversize_and_scale(viewport: RID, oversize: ViewportSDFOversize, scale: ViewportSDFScale)
void	viewport_set_size(viewport: RID, width: int, height: int)
void	viewport_set_snap_2d_transforms_to_pixel(viewport: RID, enabled: bool)
void	viewport_set_snap_2d_vertices_to_pixel(viewport: RID, enabled: bool)
void	viewport_set_texture_mipmap_bias(viewport: RID, mipmap_bias: float)
void	viewport_set_transparent_background(viewport: RID, enabled: bool)

void	viewport_set_update_mode(viewport: RID, update_mode: ViewportUpdateMode)
void	viewport_set_use_debanding(viewport: RID, enable: bool)
void	viewport_set_use_hdr_2d(viewport: RID, enabled: bool)
void	viewport_set_use_occlusion_culling(viewport: RID, enable: bool)
void	viewport_set_use_taa(viewport: RID, enable: bool)
void	viewport_set_use_xr(viewport: RID, use_xr: bool)
void	viewport_set_vrs_mode(viewport: RID, mode: ViewportVRSMode)
void	viewport_set_vrs_texture(viewport: RID, texture: RID)
void	viewport_set_vrs_update_mode(viewport: RID, mode: ViewportVRSUpdateMode)
RID	visibility_notifier_create()
void	visibility_notifier_set_aabb(notifier: RID, aabb: AABB)
void	visibility_notifier_set_callbacks(notifier: RID, enter_callable: Callable, exit_callable: Callable)
void	voxel_gi_allocate_data(voxel_gi: RID, to_cell_xform: Transform3D, aabb: AABB, octree_size: Vector3i, octree_cells: PackedByteArray, data_cells: PackedByteArray, distance_field: PackedByteArray, level_counts: PackedInt32Array)
RID	voxel_gi_create()
PackedByteArray	voxel_gi_get_data_cells(voxel_gi: RID) const
PackedByteArray	voxel_gi_get_distance_field(voxel_gi: RID) const

PackedInt32Array	voxel_gi_get_level_counts(voxel_gi: RID) const
PackedByteArray	voxel_gi_get_octree_cells(voxel_gi: RID) const
Vector3i	voxel_gi_get_octree_size(voxel_gi: RID) const
Transform3D	voxel_gi_get_to_cell_xform(voxel_gi: RID) const
void	voxel_gi_set_baked_exposure_normalization(voxel_gi: RID, baked_exposure: float)
void	voxel_gi_set_bias(voxel_gi: RID, bias: float)
void	voxel_gi_set_dynamic_range(voxel_gi: RID, range: float)
void	voxel_gi_set_energy(voxel_gi: RID, energy: float)
void	voxel_gi_set_interior(voxel_gi: RID, enable: bool)
void	voxel_gi_set_normal_bias(voxel_gi: RID, bias: float)
void	voxel_gi_set_propagation(voxel_gi: RID, amount: float)
void	voxel_gi_set_quality(quality: VoxelGIQuality)
void	voxel_gi_set_use_two_bounces(voxel_gi: RID, enable: bool)

Signals

frame_post_draw() 🔗

Emitted at the end of the frame, after the RenderingServer has finished updating all the Viewports.

frame_pre_draw() 🔗

Emitted at the beginning of the frame, before the RenderingServer updates all the Viewports.

Enumerations

enum **TextureLayeredType**: 🔗

TextureLayeredType **TEXTURE_LAYERED_2D_ARRAY** = `0`

Array of 2-dimensional textures (see Texture2DArray).

TextureLayeredType **TEXTURE_LAYERED_CUBEMAP** = `1`

Cubemap texture (see Cubemap).

TextureLayeredType **TEXTURE_LAYERED_CUBEMAP_ARRAY** = `2`

Array of cubemap textures (see CubemapArray).

---

enum **CubeMapLayer**: 🔗

CubeMapLayer **CUBEMAP_LAYER_LEFT** = `0`

Left face of a Cubemap.

CubeMapLayer **CUBEMAP_LAYER_RIGHT** = `1`

Right face of a Cubemap.

CubeMapLayer **CUBEMAP_LAYER_BOTTOM** = `2`

Bottom face of a Cubemap.

CubeMapLayer **CUBEMAP_LAYER_TOP** = `3`

Top face of a Cubemap.

CubeMapLayer **CUBEMAP_LAYER_FRONT** = `4`

Front face of a Cubemap.

CubeMapLayer **CUBEMAP_LAYER_BACK** = `5`

Back face of a Cubemap.

---

enum **ShaderMode**: 🔗

ShaderMode **SHADER_SPATIAL** = `0`

Shader is a 3D shader.

ShaderMode **SHADER_CANVAS_ITEM** = `1`

Shader is a 2D shader.

ShaderMode **SHADER_PARTICLES** = `2`

Shader is a particle shader (can be used in both 2D and 3D).

ShaderMode **SHADER_SKY** = `3`

Shader is a 3D sky shader.

ShaderMode **SHADER_FOG** = `4`

Shader is a 3D fog shader.

ShaderMode **SHADER_MAX** = `5`

Represents the size of the ShaderMode enum.

enum **ArrayType**: 🔗

ArrayType **ARRAY_VERTEX** = `0`

Array is a vertex position array.

ArrayType **ARRAY_NORMAL** = `1`

Array is a normal array.

ArrayType **ARRAY_TANGENT** = `2`

Array is a tangent array.

ArrayType **ARRAY_COLOR** = `3`

Array is a vertex color array.

ArrayType **ARRAY_TEX_UV** = `4`

Array is a UV coordinates array.

ArrayType **ARRAY_TEX_UV2** = `5`

Array is a UV coordinates array for the second set of UV coordinates.

ArrayType **ARRAY_CUSTOM0** = `6`

Array is a custom data array for the first set of custom data.

ArrayType **ARRAY_CUSTOM1** = `7`

Array is a custom data array for the second set of custom data.

ArrayType **ARRAY_CUSTOM2** = `8`

Array is a custom data array for the third set of custom data.

ArrayType **ARRAY_CUSTOM3** = `9`

Array is a custom data array for the fourth set of custom data.

ArrayType **ARRAY_BONES** = `10`

Array contains bone information.

ArrayType **ARRAY_WEIGHTS** = `11`

Array is weight information.

ArrayType **ARRAY_INDEX** = `12`

Array is an index array.

ArrayType **ARRAY_MAX** = `13`

Represents the size of the ArrayType enum.

enum **ArrayCustomFormat**: 🔗

ArrayCustomFormat **ARRAY_CUSTOM_RGBA8_UNORM** = `0`

Custom data array contains 8-bit-per-channel red/green/blue/alpha color data. Values are normalized, unsigned floating-point in the `[0.0, 1.0]` range.

ArrayCustomFormat **ARRAY_CUSTOM_RGBA8_SNORM** = `1`

Custom data array contains 8-bit-per-channel red/green/blue/alpha color data. Values are normalized, signed floating-point in the `[-1.0, 1.0]` range.

ArrayCustomFormat **ARRAY_CUSTOM_RG_HALF** = `2`

Custom data array contains 16-bit-per-channel red/green color data. Values are floating-point in half precision.

ArrayCustomFormat **ARRAY_CUSTOM_RGBA_HALF** = `3`

Custom data array contains 16-bit-per-channel red/green/blue/alpha color data. Values are floating-point in half precision.

ArrayCustomFormat **ARRAY_CUSTOM_R_FLOAT** = `4`

Custom data array contains 32-bit-per-channel red color data. Values are floating-point in single precision.

ArrayCustomFormat **ARRAY_CUSTOM_RG_FLOAT** = `5`

Custom data array contains 32-bit-per-channel red/green color data. Values are floating-point in single precision.

ArrayCustomFormat **ARRAY_CUSTOM_RGB_FLOAT** = `6`

Custom data array contains 32-bit-per-channel red/green/blue color data. Values are floating-point in single precision.

ArrayCustomFormat **ARRAY_CUSTOM_RGBA_FLOAT** = `7`

Custom data array contains 32-bit-per-channel red/green/blue/alpha color data. Values are floating-point in single precision.

ArrayCustomFormat **ARRAY_CUSTOM_MAX** = `8`

Represents the size of the ArrayCustomFormat enum.

---

flags **ArrayFormat**:

ArrayFormat **ARRAY_FORMAT_VERTEX** = `1`

Flag used to mark a vertex position array.

ArrayFormat **ARRAY_FORMAT_NORMAL** = `2`

Flag used to mark a normal array.

ArrayFormat **ARRAY_FORMAT_TANGENT** = `4`

Flag used to mark a tangent array.

ArrayFormat **ARRAY_FORMAT_COLOR** = `8`

Flag used to mark a vertex color array.

ArrayFormat **ARRAY_FORMAT_TEX_UV** = `16`

Flag used to mark a UV coordinates array.

ArrayFormat **ARRAY_FORMAT_TEX_UV2** = `32`

Flag used to mark a UV coordinates array for the second UV coordinates.

ArrayFormat **ARRAY_FORMAT_CUSTOM0** = `64`

Flag used to mark an array of custom per-vertex data for the first set of custom data.

ArrayFormat **ARRAY_FORMAT_CUSTOM1** = `128`

Flag used to mark an array of custom per-vertex data for the second set of custom data.

ArrayFormat **ARRAY_FORMAT_CUSTOM2** = `256`

Flag used to mark an array of custom per-vertex data for the third set of custom data.

ArrayFormat **ARRAY_FORMAT_CUSTOM3** = `512`

Flag used to mark an array of custom per-vertex data for the fourth set of custom data.

ArrayFormat **ARRAY_FORMAT_BONES** = `1024`

Flag used to mark a bone information array.

ArrayFormat **ARRAY_FORMAT_WEIGHTS** = `2048`

Flag used to mark a weights array.

ArrayFormat **ARRAY_FORMAT_INDEX** = `4096`

Flag used to mark an index array.

ArrayFormat **ARRAY_FORMAT_BLEND_SHAPE_MASK** = `7`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_FORMAT_CUSTOM_BASE** = `13`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_FORMAT_CUSTOM_BITS** = `3`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_FORMAT_CUSTOM0_SHIFT** = `13`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_FORMAT_CUSTOM1_SHIFT** = `16`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_FORMAT_CUSTOM2_SHIFT** = `19`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_FORMAT_CUSTOM3_SHIFT** = `22`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_FORMAT_CUSTOM_MASK** = `7`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_COMPRESS_FLAGS_BASE** = `25`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_FLAG_USE_2D_VERTICES** = `33554432`

Flag used to mark that the array contains 2D vertices.

ArrayFormat **ARRAY_FLAG_USE_DYNAMIC_UPDATE** = `67108864`

There is currently no description for this enum. Please help us by contributing one!

ArrayFormat **ARRAY_FLAG_USE_8_BONE_WEIGHTS** = `134217728`

Flag used to mark that the array uses 8 bone weights instead of 4.

ArrayFormat **ARRAY_FLAG_USES_EMPTY_VERTEX_ARRAY** = `268435456`

Flag used to mark that the mesh does not have a vertex array and instead will infer vertex positions in the shader using indices and other information.

ArrayFormat **ARRAY_FLAG_COMPRESS_ATTRIBUTES** = `536870912`

Flag used to mark that a mesh is using compressed attributes (vertices, normals, tangents, UVs). When this form of compression is enabled, vertex positions will be packed into an RGBA16UNORM attribute and scaled in the vertex shader. The normal and tangent will be packed into an RG16UNORM representing an axis, and a 16-bit float stored in the A-channel of the vertex. UVs will use 16-bit normalized floats instead of full 32-bit signed floats. When using this compression mode you must use either vertices, normals, and tangents or only vertices. You cannot use normals without tangents. Importers will automatically enable this compression if they can.

ArrayFormat **ARRAY_FLAG_FORMAT_VERSION_BASE** = `35`

Flag used to mark the start of the bits used to store the mesh version.

ArrayFormat **ARRAY_FLAG_FORMAT_VERSION_SHIFT** = `35`

Flag used to shift a mesh format int to bring the version into the lowest digits.

ArrayFormat **ARRAY_FLAG_FORMAT_VERSION_1** = `0`

Flag used to record the format used by prior mesh versions before the introduction of a version.

ArrayFormat **ARRAY_FLAG_FORMAT_VERSION_2** = `34359738368`

Flag used to record the second iteration of the mesh version flag. The primary difference between this and ARRAY_FLAG_FORMAT_VERSION_1 is that this version supports ARRAY_FLAG_COMPRESS_ATTRIBUTES and in this version vertex positions are de-interleaved from normals and tangents.

ArrayFormat **ARRAY_FLAG_FORMAT_CURRENT_VERSION** = `34359738368`

Flag used to record the current version that the engine expects. Currently this is the same as ARRAY_FLAG_FORMAT_VERSION_2.

ArrayFormat **ARRAY_FLAG_FORMAT_VERSION_MASK** = `255`

Flag used to isolate the bits used for mesh version after using ARRAY_FLAG_FORMAT_VERSION_SHIFT to shift them into place.

---

enum **PrimitiveType**: 🔗

PrimitiveType **PRIMITIVE_POINTS** = `0`

Primitive to draw consists of points.

PrimitiveType **PRIMITIVE_LINES** = `1`

Primitive to draw consists of lines.

PrimitiveType **PRIMITIVE_LINE_STRIP** = `2`

Primitive to draw consists of a line strip from start to end.

PrimitiveType **PRIMITIVE_TRIANGLES** = `3`

Primitive to draw consists of triangles.

PrimitiveType **PRIMITIVE_TRIANGLE_STRIP** = `4`

Primitive to draw consists of a triangle strip (the last 3 vertices are always combined to make a triangle).

PrimitiveType **PRIMITIVE_MAX** = `5`

Represents the size of the PrimitiveType enum.

---

enum **BlendShapeMode**: 🔗

BlendShapeMode **BLEND_SHAPE_MODE_NORMALIZED** = `0`

Blend shapes are normalized.

BlendShapeMode **BLEND_SHAPE_MODE_RELATIVE** = `1`

Blend shapes are relative to base weight.

---

enum **MultimeshTransformFormat**: 🔗

MultimeshTransformFormat **MULTIMESH_TRANSFORM_2D** = `0`

Use Transform2D to store MultiMesh transform.

MultimeshTransformFormat **MULTIMESH_TRANSFORM_3D** = `1`

Use Transform3D to store MultiMesh transform.

---

enum **LightProjectorFilter**: 🔗

LightProjectorFilter **LIGHT_PROJECTOR_FILTER_NEAREST** = `0`

Nearest-neighbor filter for light projectors (use for pixel art light projectors). No mipmaps are used for rendering, which means light projectors at a distance will look sharp but grainy. This has roughly the same performance cost as using mipmaps.

LightProjectorFilter **LIGHT_PROJECTOR_FILTER_LINEAR** = `1`

Linear filter for light projectors (use for non-pixel art light projectors). No mipmaps are used for rendering, which means light projectors at a distance will look smooth but blurry. This has roughly the same performance cost as using mipmaps.

LightProjectorFilter **LIGHT_PROJECTOR_FILTER_NEAREST_MIPMAPS** = `2`

Nearest-neighbor filter for light projectors (use for pixel art light projectors). Isotropic mipmaps are used for rendering, which means light projectors at a distance will look smooth but blurry. This has roughly the same performance cost as not using mipmaps.

LightProjectorFilter **LIGHT_PROJECTOR_FILTER_LINEAR_MIPMAPS** = `3`

Linear filter for light projectors (use for non-pixel art light projectors). Isotropic mipmaps are used for rendering, which means light projectors at a distance will look smooth but blurry. This has roughly the same performance cost as not using mipmaps.

LightProjectorFilter **LIGHT_PROJECTOR_FILTER_NEAREST_MIPMAPS_ANISOTROPIC** = `4`

Nearest-neighbor filter for light projectors (use for pixel art light projectors). Anisotropic mipmaps are used for rendering, which means light projectors at a distance will look smooth and sharp when viewed from oblique angles. This looks better compared to isotropic mipmaps, but is slower. The level of anisotropic filtering is defined by ProjectSettings.rendering/textures/default_filters/anisotropic_filtering_level.

LightProjectorFilter **LIGHT_PROJECTOR_FILTER_LINEAR_MIPMAPS_ANISOTROPIC** = `5`

Linear filter for light projectors (use for non-pixel art light projectors). Anisotropic mipmaps are used for rendering, which means light projectors at a distance will look smooth and sharp when viewed from oblique angles. This looks better compared to isotropic mipmaps, but is slower. The level of anisotropic filtering is defined by ProjectSettings.rendering/textures/default_filters/anisotropic_filtering_level.

---

enum **LightType**: 🔗

LightType **LIGHT_DIRECTIONAL** = `0`

Directional (sun/moon) light (see DirectionalLight3D).

LightType **LIGHT_OMNI** = `1`

Omni light (see OmniLight3D).

LightType **LIGHT_SPOT** = `2`

Spot light (see SpotLight3D).

---

enum **LightParam**: 🔗

LightParam **LIGHT_PARAM_ENERGY** = `0`

The light's energy multiplier.

LightParam **LIGHT_PARAM_INDIRECT_ENERGY** = `1`

The light's indirect energy multiplier (final indirect energy is LIGHT_PARAM_ENERGY * LIGHT_PARAM_INDIRECT_ENERGY).

LightParam **LIGHT_PARAM_VOLUMETRIC_FOG_ENERGY** = `2`

The light's volumetric fog energy multiplier (final volumetric fog energy is LIGHT_PARAM_ENERGY * LIGHT_PARAM_VOLUMETRIC_FOG_ENERGY).

LightParam **LIGHT_PARAM_SPECULAR** = `3`

The light's influence on specularity.

LightParam **LIGHT_PARAM_RANGE** = `4`

The light's range.

LightParam **LIGHT_PARAM_SIZE** = `5`

The size of the light when using spot light or omni light. The angular size of the light when using directional light.

LightParam **LIGHT_PARAM_ATTENUATION** = `6`

The light's attenuation.

LightParam **LIGHT_PARAM_SPOT_ANGLE** = `7`

The spotlight's angle.

LightParam **LIGHT_PARAM_SPOT_ATTENUATION** = `8`

The spotlight's attenuation.

LightParam **LIGHT_PARAM_SHADOW_MAX_DISTANCE** = `9`

The maximum distance for shadow splits. Increasing this value will make directional shadows visible from further away, at the cost of lower overall shadow detail and performance (since more objects need to be included in the directional shadow rendering).

LightParam **LIGHT_PARAM_SHADOW_SPLIT_1_OFFSET** = `10`

Proportion of shadow atlas occupied by the first split.

LightParam **LIGHT_PARAM_SHADOW_SPLIT_2_OFFSET** = `11`

Proportion of shadow atlas occupied by the second split.

LightParam **LIGHT_PARAM_SHADOW_SPLIT_3_OFFSET** = `12`

Proportion of shadow atlas occupied by the third split. The fourth split occupies the rest.

LightParam **LIGHT_PARAM_SHADOW_FADE_START** = `13`

Proportion of shadow max distance where the shadow will start to fade out.

LightParam **LIGHT_PARAM_SHADOW_NORMAL_BIAS** = `14`

Normal bias used to offset shadow lookup by object normal. Can be used to fix self-shadowing artifacts.

LightParam **LIGHT_PARAM_SHADOW_BIAS** = `15`

Bias the shadow lookup to fix self-shadowing artifacts.

LightParam **LIGHT_PARAM_SHADOW_PANCAKE_SIZE** = `16`

Sets the size of the directional shadow pancake. The pancake offsets the start of the shadow's camera frustum to provide a higher effective depth resolution for the shadow. However, a high pancake size can cause artifacts in the shadows of large objects that are close to the edge of the frustum. Reducing the pancake size can help. Setting the size to `0` turns off the pancaking effect.

LightParam **LIGHT_PARAM_SHADOW_OPACITY** = `17`

The light's shadow opacity. Values lower than `1.0` make the light appear through shadows. This can be used to fake global illumination at a low performance cost.

LightParam **LIGHT_PARAM_SHADOW_BLUR** = `18`

Blurs the edges of the shadow. Can be used to hide pixel artifacts in low resolution shadow maps. A high value can make shadows appear grainy and can cause other unwanted artifacts. Try to keep as near default as possible.

LightParam **LIGHT_PARAM_TRANSMITTANCE_BIAS** = `19`

There is currently no description for this enum. Please help us by contributing one!

LightParam **LIGHT_PARAM_INTENSITY** = `20`

Constant representing the intensity of the light, measured in Lumens when dealing with a SpotLight3D or OmniLight3D, or measured in Lux with a DirectionalLight3D. Only used when ProjectSettings.rendering/lights_and_shadows/use_physical_light_units is `true`.

LightParam **LIGHT_PARAM_MAX** = `21`

Represents the size of the LightParam enum.

---

enum **LightBakeMode**: 🔗

LightBakeMode **LIGHT_BAKE_DISABLED** = `0`

Light is ignored when baking. This is the fastest mode, but the light will be taken into account when baking global illumination. This mode should generally be used for dynamic lights that change quickly, as the effect of global illumination is less noticeable on those lights.

LightBakeMode **LIGHT_BAKE_STATIC** = `1`

Light is taken into account in static baking (VoxelGI, LightmapGI, SDFGI (Environment.sdfgi_enabled)). The light can be moved around or modified, but its global illumination will not update in real-time. This is suitable for subtle changes (such as flickering torches), but generally not large changes such as toggling a light on and off.

LightBakeMode **LIGHT_BAKE_DYNAMIC** = `2`

Light is taken into account in dynamic baking (VoxelGI and SDFGI (Environment.sdfgi_enabled) only). The light can be moved around or modified with global illumination updating in real-time. The light's global illumination appearance will be slightly different compared to LIGHT_BAKE_STATIC. This has a greater performance cost compared to LIGHT_BAKE_STATIC. When using SDFGI, the update speed of dynamic lights is affected by ProjectSettings.rendering/global_illumination/sdfgi/frames_to_update_lights.

enum **LightOmniShadowMode**: 🔗

LightOmniShadowMode **LIGHT_OMNI_SHADOW_DUAL_PARABOLOID** = `0`

Use a dual paraboloid shadow map for omni lights.

LightOmniShadowMode **LIGHT_OMNI_SHADOW_CUBE** = `1`

Use a cubemap shadow map for omni lights. Slower but better quality than dual paraboloid.

enum **LightDirectionalShadowMode**: 🔗

LightDirectionalShadowMode **LIGHT_DIRECTIONAL_SHADOW_ORTHOGONAL** = `0`

Use orthogonal shadow projection for directional light.

LightDirectionalShadowMode **LIGHT_DIRECTIONAL_SHADOW_PARALLEL_2_SPLITS** = `1`

Use 2 splits for shadow projection when using directional light.

LightDirectionalShadowMode **LIGHT_DIRECTIONAL_SHADOW_PARALLEL_4_SPLITS** = `2`

Use 4 splits for shadow projection when using directional light.

enum **LightDirectionalSkyMode**: 🔗

LightDirectionalSkyMode **LIGHT_DIRECTIONAL_SKY_MODE_LIGHT_AND_SKY** = `0`

Use DirectionalLight3D in both sky rendering and scene lighting.

LightDirectionalSkyMode **LIGHT_DIRECTIONAL_SKY_MODE_LIGHT_ONLY** = `1`

Only use DirectionalLight3D in scene lighting.

LightDirectionalSkyMode **LIGHT_DIRECTIONAL_SKY_MODE_SKY_ONLY** = `2`

Only use DirectionalLight3D in sky rendering.

enum **ShadowQuality**: 🔗

ShadowQuality **SHADOW_QUALITY_HARD** = `0`

Lowest shadow filtering quality (fastest). Soft shadows are not available with this quality setting, which means the Light3D.shadow_blur property is ignored if Light3D.light_size and

Light3D.light_angular_distance is `0.0`.

Note: The variable shadow blur performed by Light3D.light_size and Light3D.light_angular_distance is still effective when using hard shadow filtering. In this case, Light3D.shadow_blur *is* taken into account. However, the results will not be blurred, instead the blur amount is treated as a maximum radius for the penumbra.

ShadowQuality **SHADOW_QUALITY_SOFT_VERY_LOW** = `1`

Very low shadow filtering quality (faster). When using this quality setting, Light3D.shadow_blur is automatically multiplied by 0.75× to avoid introducing too much noise. This division only applies to lights whose Light3D.light_size or Light3D.light_angular_distance is `0.0`).

ShadowQuality **SHADOW_QUALITY_SOFT_LOW** = `2`

Low shadow filtering quality (fast).

ShadowQuality **SHADOW_QUALITY_SOFT_MEDIUM** = `3`

Medium low shadow filtering quality (average).

ShadowQuality **SHADOW_QUALITY_SOFT_HIGH** = `4`

High low shadow filtering quality (slow). When using this quality setting, Light3D.shadow_blur is automatically multiplied by 1.5× to better make use of the high sample count. This increased blur also improves the stability of dynamic object shadows. This multiplier only applies to lights whose Light3D.light_size or Light3D.light_angular_distance is `0.0`).

ShadowQuality **SHADOW_QUALITY_SOFT_ULTRA** = `5`

Highest low shadow filtering quality (slowest). When using this quality setting, Light3D.shadow_blur is automatically multiplied by 2× to better make use of the high sample count. This increased blur also improves the stability of dynamic object shadows. This multiplier only applies to lights whose Light3D.light_size or Light3D.light_angular_distance is `0.0`).

ShadowQuality **SHADOW_QUALITY_MAX** = `6`

Represents the size of the ShadowQuality enum.

enum **ReflectionProbeUpdateMode**:

ReflectionProbeUpdateMode **REFLECTION_PROBE_UPDATE_ONCE** = `0`

Reflection probe will update reflections once and then stop.

ReflectionProbeUpdateMode **REFLECTION_PROBE_UPDATE_ALWAYS** = `1`

Reflection probe will update each frame. This mode is necessary to capture moving objects.

enum **ReflectionProbeAmbientMode**:

ReflectionProbeAmbientMode **REFLECTION_PROBE_AMBIENT_DISABLED** = `0`

Do not apply any ambient lighting inside the reflection probe's box defined by its size.

ReflectionProbeAmbientMode **REFLECTION_PROBE_AMBIENT_ENVIRONMENT** = `1`

Apply automatically-sourced environment lighting inside the reflection probe's box defined by its size.

ReflectionProbeAmbientMode **REFLECTION_PROBE_AMBIENT_COLOR** = `2`

Apply custom ambient lighting inside the reflection probe's box defined by its size. See reflection_probe_set_ambient_color and reflection_probe_set_ambient_energy.

enum **DecalTexture**: 🔗

DecalTexture **DECAL_TEXTURE_ALBEDO** = `0`

Albedo texture slot in a decal (Decal.texture_albedo).

DecalTexture **DECAL_TEXTURE_NORMAL** = `1`

Normal map texture slot in a decal (Decal.texture_normal).

DecalTexture **DECAL_TEXTURE_ORM** = `2`

Occlusion/Roughness/Metallic texture slot in a decal (Decal.texture_orm).

DecalTexture **DECAL_TEXTURE_EMISSION** = `3`

Emission texture slot in a decal (Decal.texture_emission).

DecalTexture **DECAL_TEXTURE_MAX** = `4`

Represents the size of the DecalTexture enum.

enum **DecalFilter**: 🔗

DecalFilter **DECAL_FILTER_NEAREST** = `0`

Nearest-neighbor filter for decals (use for pixel art decals). No mipmaps are used for rendering, which means decals at a distance will look sharp but grainy. This has roughly the same performance cost as using mipmaps.

DecalFilter **DECAL_FILTER_LINEAR** = `1`

Linear filter for decals (use for non-pixel art decals). No mipmaps are used for rendering, which means decals at a distance will look smooth but blurry. This has roughly the same performance cost as using mipmaps.

DecalFilter **DECAL_FILTER_NEAREST_MIPMAPS** = `2`

Nearest-neighbor filter for decals (use for pixel art decals). Isotropic mipmaps are used for rendering, which means decals at a distance will look smooth but blurry. This has roughly the same performance cost as not using mipmaps.

DecalFilter **DECAL_FILTER_LINEAR_MIPMAPS** = `3`

Linear filter for decals (use for non-pixel art decals). Isotropic mipmaps are used for rendering, which means decals at a distance will look smooth but blurry. This has roughly the same performance cost as not using mipmaps.

DecalFilter **DECAL_FILTER_NEAREST_MIPMAPS_ANISOTROPIC** = `4`

Nearest-neighbor filter for decals (use for pixel art decals). Anisotropic mipmaps are used for rendering, which means decals at a distance will look smooth and sharp when viewed from oblique angles. This looks better compared to isotropic mipmaps, but is slower. The level of anisotropic filtering is defined by ProjectSettings.rendering/textures/default_filters/anisotropic_filtering_level.

DecalFilter **DECAL_FILTER_LINEAR_MIPMAPS_ANISOTROPIC** = `5`

Linear filter for decals (use for non-pixel art decals). Anisotropic mipmaps are used for rendering, which means decals at a distance will look smooth and sharp when viewed from oblique angles. This looks better compared to isotropic mipmaps, but is slower. The level of anisotropic filtering is defined by ProjectSettings.rendering/textures/default_filters/anisotropic_filtering_level.

---

enum **VoxelGIQuality**:

VoxelGIQuality **VOXEL_GI_QUALITY_LOW** = `0`

Low VoxelGI rendering quality using 4 cones.

VoxelGIQuality **VOXEL_GI_QUALITY_HIGH** = `1`

High VoxelGI rendering quality using 6 cones.

---

enum **ParticlesMode**:

ParticlesMode **PARTICLES_MODE_2D** = `0`

2D particles.

ParticlesMode **PARTICLES_MODE_3D** = `1`

3D particles.

---

enum **ParticlesTransformAlign**:

ParticlesTransformAlign **PARTICLES_TRANSFORM_ALIGN_DISABLED** = `0`

There is currently no description for this enum. Please help us by contributing one!

ParticlesTransformAlign **PARTICLES_TRANSFORM_ALIGN_Z_BILLBOARD** = `1`

There is currently no description for this enum. Please help us by contributing one!

ParticlesTransformAlign **PARTICLES_TRANSFORM_ALIGN_Y_TO_VELOCITY** = `2`

There is currently no description for this enum. Please help us by contributing one!

ParticlesTransformAlign **PARTICLES_TRANSFORM_ALIGN_Z_BILLBOARD_Y_TO_VELOCITY** = `3`

There is currently no description for this enum. Please help us by contributing one!

---

enum **ParticlesDrawOrder**: 🔗

ParticlesDrawOrder **PARTICLES_DRAW_ORDER_INDEX** = `0`

Draw particles in the order that they appear in the particles array.

ParticlesDrawOrder **PARTICLES_DRAW_ORDER_LIFETIME** = `1`

Sort particles based on their lifetime. In other words, the particle with the highest lifetime is drawn at the front.

ParticlesDrawOrder **PARTICLES_DRAW_ORDER_REVERSE_LIFETIME** = `2`

Sort particles based on the inverse of their lifetime. In other words, the particle with the lowest lifetime is drawn at the front.

ParticlesDrawOrder **PARTICLES_DRAW_ORDER_VIEW_DEPTH** = `3`

Sort particles based on their distance to the camera.

---

enum **ParticlesCollisionType**: 🔗

ParticlesCollisionType **PARTICLES_COLLISION_TYPE_SPHERE_ATTRACT** = `0`

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionType **PARTICLES_COLLISION_TYPE_BOX_ATTRACT** = `1`

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionType **PARTICLES_COLLISION_TYPE_VECTOR_FIELD_ATTRACT** = `2`

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionType **PARTICLES_COLLISION_TYPE_SPHERE_COLLIDE** = `3`

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionType **PARTICLES_COLLISION_TYPE_BOX_COLLIDE** = `4`

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionType **PARTICLES_COLLISION_TYPE_SDF_COLLIDE** = `5`

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionType **PARTICLES_COLLISION_TYPE_HEIGHTFIELD_COLLIDE** = 6

There is currently no description for this enum. Please help us by contributing one!

---

enum **ParticlesCollisionHeightfieldResolution**: 🔗

ParticlesCollisionHeightfieldResolution
PARTICLES_COLLISION_HEIGHTFIELD_RESOLUTION_256 = 0

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionHeightfieldResolution
PARTICLES_COLLISION_HEIGHTFIELD_RESOLUTION_512 = 1

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionHeightfieldResolution
PARTICLES_COLLISION_HEIGHTFIELD_RESOLUTION_1024 = 2

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionHeightfieldResolution
PARTICLES_COLLISION_HEIGHTFIELD_RESOLUTION_2048 = 3

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionHeightfieldResolution
PARTICLES_COLLISION_HEIGHTFIELD_RESOLUTION_4096 = 4

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionHeightfieldResolution
PARTICLES_COLLISION_HEIGHTFIELD_RESOLUTION_8192 = 5

There is currently no description for this enum. Please help us by contributing one!

ParticlesCollisionHeightfieldResolution
PARTICLES_COLLISION_HEIGHTFIELD_RESOLUTION_MAX = 6

Represents the size of the ParticlesCollisionHeightfieldResolution enum.

---

enum **FogVolumeShape**: 🔗

FogVolumeShape **FOG_VOLUME_SHAPE_ELLIPSOID** = 0

FogVolume will be shaped like an ellipsoid (stretched sphere).

FogVolumeShape **FOG_VOLUME_SHAPE_CONE** = 1

FogVolume will be shaped like a cone pointing upwards (in local coordinates). The cone's angle is set automatically to fill the size. The cone will be adjusted to fit within the size. Rotate the FogVolume node to reorient the cone. Non-uniform scaling via size is not supported (scale the FogVolume node instead).

FogVolumeShape **FOG_VOLUME_SHAPE_CYLINDER** = `2`

FogVolume will be shaped like an upright cylinder (in local coordinates). Rotate the FogVolume node to reorient the cylinder. The cylinder will be adjusted to fit within the size. Non-uniform scaling via size is not supported (scale the FogVolume node instead).

FogVolumeShape **FOG_VOLUME_SHAPE_BOX** = `3`

FogVolume will be shaped like a box.

FogVolumeShape **FOG_VOLUME_SHAPE_WORLD** = `4`

FogVolume will have no shape, will cover the whole world and will not be culled.

FogVolumeShape **FOG_VOLUME_SHAPE_MAX** = `5`

Represents the size of the FogVolumeShape enum.

---

enum **ViewportScaling3DMode**: 🔗

ViewportScaling3DMode **VIEWPORT_SCALING_3D_MODE_BILINEAR** = `0`

Use bilinear scaling for the viewport's 3D buffer. The amount of scaling can be set using Viewport.scaling_3d_scale. Values less than `1.0` will result in undersampling while values greater than `1.0` will result in supersampling. A value of `1.0` disables scaling.

ViewportScaling3DMode **VIEWPORT_SCALING_3D_MODE_FSR** = `1`

Use AMD FidelityFX Super Resolution 1.0 upscaling for the viewport's 3D buffer. The amount of scaling can be set using Viewport.scaling_3d_scale. Values less than `1.0` will be result in the viewport being upscaled using FSR. Values greater than `1.0` are not supported and bilinear downsampling will be used instead. A value of `1.0` disables scaling.

ViewportScaling3DMode **VIEWPORT_SCALING_3D_MODE_FSR2** = `2`

Use AMD FidelityFX Super Resolution 2.2 upscaling for the viewport's 3D buffer. The amount of scaling can be set using Viewport.scaling_3d_scale. Values less than `1.0` will be result in the viewport being upscaled using FSR2. Values greater than `1.0` are not supported and bilinear downsampling will be used instead. A value of `1.0` will use FSR2 at native resolution as a TAA solution.

ViewportScaling3DMode **VIEWPORT_SCALING_3D_MODE_MAX** = `3`

Represents the size of the ViewportScaling3DMode enum.

---

enum **ViewportUpdateMode**: 🔗

ViewportUpdateMode **VIEWPORT_UPDATE_DISABLED** = `0`

Do not update the viewport's render target.

ViewportUpdateMode **VIEWPORT_UPDATE_ONCE** = `1`

Update the viewport's render target once, then switch to VIEWPORT_UPDATE_DISABLED.

ViewportUpdateMode **VIEWPORT_UPDATE_WHEN_VISIBLE** = `2`

Update the viewport's render target only when it is visible. This is the default value.

ViewportUpdateMode **VIEWPORT_UPDATE_WHEN_PARENT_VISIBLE** = `3`

Update the viewport's render target only when its parent is visible.

ViewportUpdateMode **VIEWPORT_UPDATE_ALWAYS** = `4`

Always update the viewport's render target.

---

enum **ViewportClearMode**:

ViewportClearMode **VIEWPORT_CLEAR_ALWAYS** = `0`

Always clear the viewport's render target before drawing.

ViewportClearMode **VIEWPORT_CLEAR_NEVER** = `1`

Never clear the viewport's render target.

ViewportClearMode **VIEWPORT_CLEAR_ONLY_NEXT_FRAME** = `2`

Clear the viewport's render target on the next frame, then switch to VIEWPORT_CLEAR_NEVER.

---

enum **ViewportEnvironmentMode**:

ViewportEnvironmentMode **VIEWPORT_ENVIRONMENT_DISABLED** = `0`

Disable rendering of 3D environment over 2D canvas.

ViewportEnvironmentMode **VIEWPORT_ENVIRONMENT_ENABLED** = `1`

Enable rendering of 3D environment over 2D canvas.

ViewportEnvironmentMode **VIEWPORT_ENVIRONMENT_INHERIT** = `2`

Inherit enable/disable value from parent. If the topmost parent is also set to VIEWPORT_ENVIRONMENT_INHERIT, then this has the same behavior as VIEWPORT_ENVIRONMENT_ENABLED.

ViewportEnvironmentMode **VIEWPORT_ENVIRONMENT_MAX** = `3`

Represents the size of the ViewportEnvironmentMode enum.

---

enum **ViewportSDFOversize**:

ViewportSDFOversize **VIEWPORT_SDF_OVERSIZE_100_PERCENT** = `0`

Do not oversize the 2D signed distance field. Occluders may disappear when touching the viewport's edges, and GPUParticles3D collision may stop working earlier than intended. This has the lowest GPU requirements.

ViewportSDFOversize **VIEWPORT_SDF_OVERSIZE_120_PERCENT** = `1`

2D signed distance field covers 20% of the viewport's size outside the viewport on each side (top, right, bottom, left).

ViewportSDFOversize **VIEWPORT_SDF_OVERSIZE_150_PERCENT** = `2`

2D signed distance field covers 50% of the viewport's size outside the viewport on each side (top, right, bottom, left).

ViewportSDFOversize **VIEWPORT_SDF_OVERSIZE_200_PERCENT** = `3`

2D signed distance field covers 100% of the viewport's size outside the viewport on each side (top, right, bottom, left). This has the highest GPU requirements.

ViewportSDFOversize **VIEWPORT_SDF_OVERSIZE_MAX** = `4`

Represents the size of the ViewportSDFOversize enum.

---

enum **ViewportSDFScale**:

ViewportSDFScale **VIEWPORT_SDF_SCALE_100_PERCENT** = `0`

Full resolution 2D signed distance field scale. This has the highest GPU requirements.

ViewportSDFScale **VIEWPORT_SDF_SCALE_50_PERCENT** = `1`

Half resolution 2D signed distance field scale on each axis (25% of the viewport pixel count).

ViewportSDFScale **VIEWPORT_SDF_SCALE_25_PERCENT** = `2`

Quarter resolution 2D signed distance field scale on each axis (6.25% of the viewport pixel count). This has the lowest GPU requirements.

ViewportSDFScale **VIEWPORT_SDF_SCALE_MAX** = `3`

Represents the size of the ViewportSDFScale enum.

---

enum **ViewportMSAA**:

ViewportMSAA **VIEWPORT_MSAA_DISABLED** = `0`

Multisample antialiasing for 3D is disabled. This is the default value, and also the fastest setting.

ViewportMSAA **VIEWPORT_MSAA_2X** = `1`

Multisample antialiasing uses 2 samples per pixel for 3D. This has a moderate impact on performance.

ViewportMSAA **VIEWPORT_MSAA_4X** = `2`

Multisample antialiasing uses 4 samples per pixel for 3D. This has a high impact on performance.

ViewportMSAA **VIEWPORT_MSAA_8X** = `3`

Multisample antialiasing uses 8 samples per pixel for 3D. This has a very high impact on performance. Likely unsupported on low-end and older hardware.

ViewportMSAA **VIEWPORT_MSAA_MAX** = `4`

Represents the size of the ViewportMSAA enum.

enum **ViewportScreenSpaceAA**: 🔗

ViewportScreenSpaceAA **VIEWPORT_SCREEN_SPACE_AA_DISABLED** = `0`

Do not perform any antialiasing in the full screen post-process.

ViewportScreenSpaceAA **VIEWPORT_SCREEN_SPACE_AA_FXAA** = `1`

Use fast approximate antialiasing. FXAA is a popular screen-space antialiasing method, which is fast but will make the image look blurry, especially at lower resolutions. It can still work relatively well at large resolutions such as 1440p and 4K.

ViewportScreenSpaceAA **VIEWPORT_SCREEN_SPACE_AA_MAX** = `2`

Represents the size of the ViewportScreenSpaceAA enum.

enum **ViewportOcclusionCullingBuildQuality**: 🔗

ViewportOcclusionCullingBuildQuality **VIEWPORT_OCCLUSION_BUILD_QUALITY_LOW** = `0`

Low occlusion culling BVH build quality (as defined by Embree). Results in the lowest CPU usage, but least effective culling.

ViewportOcclusionCullingBuildQuality **VIEWPORT_OCCLUSION_BUILD_QUALITY_MEDIUM** = `1`

Medium occlusion culling BVH build quality (as defined by Embree).

ViewportOcclusionCullingBuildQuality **VIEWPORT_OCCLUSION_BUILD_QUALITY_HIGH** = `2`

High occlusion culling BVH build quality (as defined by Embree). Results in the highest CPU usage, but most effective culling.

enum **ViewportRenderInfo**: 🔗

ViewportRenderInfo **VIEWPORT_RENDER_INFO_OBJECTS_IN_FRAME** = `0`

Number of objects drawn in a single frame.

ViewportRenderInfo **VIEWPORT_RENDER_INFO_PRIMITIVES_IN_FRAME** = `1`

Number of points, lines, or triangles drawn in a single frame.

ViewportRenderInfo **VIEWPORT_RENDER_INFO_DRAW_CALLS_IN_FRAME** = `2`

Number of draw calls during this frame.

ViewportRenderInfo **VIEWPORT_RENDER_INFO_MAX** = `3`

Represents the size of the ViewportRenderInfo enum.

enum **ViewportRenderInfoType**: 🔗

ViewportRenderInfoType **VIEWPORT_RENDER_INFO_TYPE_VISIBLE** = `0`

Visible render pass (excluding shadows).

ViewportRenderInfoType **VIEWPORT_RENDER_INFO_TYPE_SHADOW** = `1`

Shadow render pass. Objects will be rendered several times depending on the number of amounts of lights with shadows and the number of directional shadow splits.

ViewportRenderInfoType **VIEWPORT_RENDER_INFO_TYPE_CANVAS** = `2`

Canvas item rendering. This includes all 2D rendering.

ViewportRenderInfoType **VIEWPORT_RENDER_INFO_TYPE_MAX** = `3`

Represents the size of the ViewportRenderInfoType enum.

enum **ViewportDebugDraw**: 🔗

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_DISABLED** = `0`

Debug draw is disabled. Default setting.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_UNSHADED** = `1`

Objects are displayed without light information.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_LIGHTING** = `2`

Objects are displayed with only light information.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_OVERDRAW** = `3`

Objects are displayed semi-transparent with additive blending so you can see where they are drawing over top of one another. A higher overdraw (represented by brighter colors) means you are wasting performance on drawing pixels that are being hidden behind others.

Note: When using this debug draw mode, custom shaders will be ignored. This means vertex displacement won't be visible anymore.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_WIREFRAME** = `4`

Debug draw draws objects in wireframe.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_NORMAL_BUFFER** = `5`

Normal buffer is drawn instead of regular scene so you can see the per-pixel normals that will be used by post-processing effects.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_VOXEL_GI_ALBEDO** = `6`

Objects are displayed with only the albedo value from VoxelGIs.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_VOXEL_GI_LIGHTING** = `7`

Objects are displayed with only the lighting value from VoxelGIs.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_VOXEL_GI_EMISSION** = `8`

Objects are displayed with only the emission color from VoxelGIs.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_SHADOW_ATLAS** = `9`

Draws the shadow atlas that stores shadows from OmniLight3Ds and SpotLight3Ds in the upper left quadrant of the Viewport.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_DIRECTIONAL_SHADOW_ATLAS** = `10`

Draws the shadow atlas that stores shadows from DirectionalLight3Ds in the upper left quadrant of the Viewport.

The slice of the camera frustum related to the shadow map cascade is superimposed to visualize coverage. The color of each slice matches the colors used for VIEWPORT_DEBUG_DRAW_PSSM_SPLITS. When shadow cascades are blended the overlap is taken into account when drawing the frustum slices.

The last cascade shows all frustum slices to illustrate the coverage of all slices.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_SCENE_LUMINANCE** = `11`

Draws the estimated scene luminance. This is a 1×1 texture that is generated when autoexposure is enabled to control the scene's exposure.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_SSAO** = `12`

Draws the screen space ambient occlusion texture instead of the scene so that you can clearly see how it is affecting objects. In order for this display mode to work, you must have Environment.ssao_enabled set in your WorldEnvironment.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_SSIL** = `13`

Draws the screen space indirect lighting texture instead of the scene so that you can clearly see how it is affecting objects. In order for this display mode to work, you must have Environment.ssil_enabled set in your WorldEnvironment.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_PSSM_SPLITS** = `14`

Colors each PSSM split for the DirectionalLight3Ds in the scene a different color so you can see where the splits are. In order they will be colored red, green, blue, yellow.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_DECAL_ATLAS** = `15`

Draws the decal atlas that stores decal textures from Decals.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_SDFGI** = `16`

Draws SDFGI cascade data. This is the data structure that is used to bounce lighting against and create reflections.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_SDFGI_PROBES** = `17`

Draws SDFGI probe data. This is the data structure that is used to give indirect lighting dynamic objects moving within the scene.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_GI_BUFFER** = `18`

Draws the global illumination buffer (VoxelGI or SDFGI).

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_DISABLE_LOD** = `19`

Disable mesh LOD. All meshes are drawn with full detail, which can be used to compare performance.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_CLUSTER_OMNI_LIGHTS** = `20`

Draws the OmniLight3D cluster. Clustering determines where lights are positioned in screen-space, which allows the engine to only process these portions of the screen for lighting.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_CLUSTER_SPOT_LIGHTS** = `21`

Draws the SpotLight3D cluster. Clustering determines where lights are positioned in screen-space, which allows the engine to only process these portions of the screen for lighting.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_CLUSTER_DECALS** = `22`

Draws the Decal cluster. Clustering determines where decals are positioned in screen-space, which allows the engine to only process these portions of the screen for decals.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_CLUSTER_REFLECTION_PROBES** = `23`

Draws the ReflectionProbe cluster. Clustering determines where reflection probes are positioned in screen-space, which allows the engine to only process these portions of the screen for reflection probes.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_OCCLUDERS** = `24`

Draws the occlusion culling buffer. This low-resolution occlusion culling buffer is rasterized on the CPU and is used to check whether instances are occluded by other objects.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_MOTION_VECTORS** = `25`

Draws the motion vectors buffer. This is used by temporal antialiasing to correct for motion that occurs during gameplay.

ViewportDebugDraw **VIEWPORT_DEBUG_DRAW_INTERNAL_BUFFER** = `26`

Internal buffer is drawn instead of regular scene so you can see the per-pixel output that will be used by post-processing effects.

---

enum **ViewportVRSMode**: 🔗

ViewportVRSMode **VIEWPORT_VRS_DISABLED** = `0`

Variable rate shading is disabled.

ViewportVRSMode **VIEWPORT_VRS_TEXTURE** = `1`

Variable rate shading uses a texture. Note, for stereoscopic use a texture atlas with a texture for each view.

ViewportVRSMode **VIEWPORT_VRS_XR** = `2`

Variable rate shading texture is supplied by the primary XRInterface. Note that this may override the update mode.

ViewportVRSMode **VIEWPORT_VRS_MAX** = `3`

Represents the size of the ViewportVRSMode enum.

---

enum **ViewportVRSUpdateMode**: 🔗

ViewportVRSUpdateMode **VIEWPORT_VRS_UPDATE_DISABLED** = `0`

The input texture for variable rate shading will not be processed.

ViewportVRSUpdateMode **VIEWPORT_VRS_UPDATE_ONCE** = `1`

The input texture for variable rate shading will be processed once.

ViewportVRSUpdateMode **VIEWPORT_VRS_UPDATE_ALWAYS** = `2`

The input texture for variable rate shading will be processed each frame.

ViewportVRSUpdateMode **VIEWPORT_VRS_UPDATE_MAX** = `3`

Represents the size of the ViewportVRSUpdateMode enum.

---

enum **SkyMode**: 🔗

SkyMode **SKY_MODE_AUTOMATIC** = `0`

Automatically selects the appropriate process mode based on your sky shader. If your shader uses `TIME` or `POSITION`, this will use SKY_MODE_REALTIME. If your shader uses any of the `LIGHT_*` variables or any custom uniforms, this uses SKY_MODE_INCREMENTAL. Otherwise, this defaults to SKY_MODE_QUALITY.

SkyMode **SKY_MODE_QUALITY** = `1`

Uses high quality importance sampling to process the radiance map. In general, this results in much higher quality than SKY_MODE_REALTIME but takes much longer to generate. This should not be used if you plan on changing the sky at runtime. If you are finding that the reflection is not blurry enough and is showing sparkles or fireflies, try increasing ProjectSettings.rendering/reflections/sky_reflections/ggx_samples.

SkyMode **SKY_MODE_INCREMENTAL** = `2`

Uses the same high quality importance sampling to process the radiance map as SKY_MODE_QUALITY, but updates over several frames. The number of frames is determined by ProjectSettings.rendering/reflections/sky_reflections/roughness_layers. Use this when you need highest quality radiance maps, but have a sky that updates slowly.

SkyMode **SKY_MODE_REALTIME** = `3`

Uses the fast filtering algorithm to process the radiance map. In general this results in lower quality, but substantially faster run times. If you need better quality, but still need to update the sky every frame, consider turning on ProjectSettings.rendering/reflections/sky_reflections/fast_filter_high_quality.

Note: The fast filtering algorithm is limited to 256×256 cubemaps, so sky_set_radiance_size must be set to `256`. Otherwise, a warning is printed and the overridden radiance size is ignored.

---

enum **CompositorEffectFlags**: 🔗

CompositorEffectFlags **COMPOSITOR_EFFECT_FLAG_ACCESS_RESOLVED_COLOR** = `1`

The rendering effect requires the color buffer to be resolved if MSAA is enabled.

CompositorEffectFlags **COMPOSITOR_EFFECT_FLAG_ACCESS_RESOLVED_DEPTH** = `2`

The rendering effect requires the depth buffer to be resolved if MSAA is enabled.

CompositorEffectFlags **COMPOSITOR_EFFECT_FLAG_NEEDS_MOTION_VECTORS** = `4`

The rendering effect requires motion vectors to be produced.

CompositorEffectFlags **COMPOSITOR_EFFECT_FLAG_NEEDS_ROUGHNESS** = `8`

The rendering effect requires normals and roughness g-buffer to be produced (Forward+ only).

CompositorEffectFlags **COMPOSITOR_EFFECT_FLAG_NEEDS_SEPARATE_SPECULAR** = `16`

The rendering effect requires specular data to be separated out (Forward+ only).

---

enum **CompositorEffectCallbackType**: 🔗

CompositorEffectCallbackType **COMPOSITOR_EFFECT_CALLBACK_TYPE_PRE_OPAQUE** = `0`

The callback is called before our opaque rendering pass, but after depth prepass (if applicable).

CompositorEffectCallbackType **COMPOSITOR_EFFECT_CALLBACK_TYPE_POST_OPAQUE** = `1`

The callback is called after our opaque rendering pass, but before our sky is rendered.

CompositorEffectCallbackType **COMPOSITOR_EFFECT_CALLBACK_TYPE_POST_SKY** = `2`

The callback is called after our sky is rendered, but before our back buffers are created (and if enabled, before subsurface scattering and/or screen space reflections).

CompositorEffectCallbackType **COMPOSITOR_EFFECT_CALLBACK_TYPE_PRE_TRANSPARENT** = `3`

The callback is called before our transparent rendering pass, but after our sky is rendered and we've created our back buffers.

CompositorEffectCallbackType **COMPOSITOR_EFFECT_CALLBACK_TYPE_POST_TRANSPARENT** = `4`

The callback is called after our transparent rendering pass, but before any build in post effects and output to our render target.

CompositorEffectCallbackType **COMPOSITOR_EFFECT_CALLBACK_TYPE_ANY** = `-1`

There is currently no description for this enum. Please help us by contributing one!

---

enum **EnvironmentBG**: 🔗

EnvironmentBG **ENV_BG_CLEAR_COLOR** = `0`

Use the clear color as background.

EnvironmentBG **ENV_BG_COLOR** = `1`

Use a specified color as the background.

EnvironmentBG **ENV_BG_SKY** = `2`

Use a sky resource for the background.

EnvironmentBG **ENV_BG_CANVAS** = `3`

Use a specified canvas layer as the background. This can be useful for instantiating a 2D scene in a 3D world.

EnvironmentBG **ENV_BG_KEEP** = `4`

Do not clear the background, use whatever was rendered last frame as the background.

EnvironmentBG **ENV_BG_CAMERA_FEED** = `5`

Displays a camera feed in the background.

EnvironmentBG **ENV_BG_MAX** = `6`

Represents the size of the EnvironmentBG enum.

---

enum **EnvironmentAmbientSource**:

EnvironmentAmbientSource **ENV_AMBIENT_SOURCE_BG** = `0`

Gather ambient light from whichever source is specified as the background.

EnvironmentAmbientSource **ENV_AMBIENT_SOURCE_DISABLED** = `1`

Disable ambient light.

EnvironmentAmbientSource **ENV_AMBIENT_SOURCE_COLOR** = `2`

Specify a specific Color for ambient light.

EnvironmentAmbientSource **ENV_AMBIENT_SOURCE_SKY** = `3`

Gather ambient light from the Sky regardless of what the background is.

---

enum **EnvironmentReflectionSource**:

EnvironmentReflectionSource **ENV_REFLECTION_SOURCE_BG** = `0`

Use the background for reflections.

EnvironmentReflectionSource **ENV_REFLECTION_SOURCE_DISABLED** = `1`

Disable reflections.

EnvironmentReflectionSource **ENV_REFLECTION_SOURCE_SKY** = `2`

Use the Sky for reflections regardless of what the background is.

---

enum **EnvironmentGlowBlendMode**:

EnvironmentGlowBlendMode **ENV_GLOW_BLEND_MODE_ADDITIVE** = `0`

Additive glow blending mode. Mostly used for particles, glows (bloom), lens flare, bright sources.

EnvironmentGlowBlendMode **ENV_GLOW_BLEND_MODE_SCREEN** = `1`

Screen glow blending mode. Increases brightness, used frequently with bloom.

EnvironmentGlowBlendMode **ENV_GLOW_BLEND_MODE_SOFTLIGHT** = `2`

Soft light glow blending mode. Modifies contrast, exposes shadows and highlights (vivid bloom).

EnvironmentGlowBlendMode **ENV_GLOW_BLEND_MODE_REPLACE** = `3`

Replace glow blending mode. Replaces all pixels' color by the glow value. This can be used to simulate a full-screen blur effect by tweaking the glow parameters to match the original image's brightness.

EnvironmentGlowBlendMode **ENV_GLOW_BLEND_MODE_MIX** = `4`

Mixes the glow with the underlying color to avoid increasing brightness as much while still maintaining a glow effect.

enum **EnvironmentFogMode**: 🔗

EnvironmentFogMode **ENV_FOG_MODE_EXPONENTIAL** = `0`

Use a physically-based fog model defined primarily by fog density.

EnvironmentFogMode **ENV_FOG_MODE_DEPTH** = `1`

Use a simple fog model defined by start and end positions and a custom curve. While not physically accurate, this model can be useful when you need more artistic control.

enum **EnvironmentToneMapper**: 🔗

EnvironmentToneMapper **ENV_TONE_MAPPER_LINEAR** = `0`

Output color as they came in. This can cause bright lighting to look blown out, with noticeable clipping in the output colors.

EnvironmentToneMapper **ENV_TONE_MAPPER_REINHARD** = `1`

Use the Reinhard tonemapper. Performs a variation on rendered pixels' colors by this formula: `color = color / (1 + color)`. This avoids clipping bright highlights, but the resulting image can look a bit dull.

EnvironmentToneMapper **ENV_TONE_MAPPER_FILMIC** = `2`

Use the filmic tonemapper. This avoids clipping bright highlights, with a resulting image that usually looks more vivid than ENV_TONE_MAPPER_REINHARD.

EnvironmentToneMapper **ENV_TONE_MAPPER_ACES** = `3`

Use the Academy Color Encoding System tonemapper. ACES is slightly more expensive than other options, but it handles bright lighting in a more realistic fashion by desaturating it as it becomes brighter. ACES typically has a more contrasted output compared to ENV_TONE_MAPPER_REINHARD and ENV_TONE_MAPPER_FILMIC.

Note: This tonemapping operator is called "ACES Fitted" in Godot 3.x.

enum **EnvironmentSSRRoughnessQuality**: 🔗

EnvironmentSSRRoughnessQuality **ENV_SSR_ROUGHNESS_QUALITY_DISABLED** = `0`

Lowest quality of roughness filter for screen-space reflections. Rough materials will not have blurrier screen-space reflections compared to smooth (non-rough) materials. This is the fastest option.

EnvironmentSSRRoughnessQuality **ENV_SSR_ROUGHNESS_QUALITY_LOW** = `1`

Low quality of roughness filter for screen-space reflections.

EnvironmentSSRRoughnessQuality **ENV_SSR_ROUGHNESS_QUALITY_MEDIUM** = `2`

Medium quality of roughness filter for screen-space reflections.

EnvironmentSSRRoughnessQuality **ENV_SSR_ROUGHNESS_QUALITY_HIGH** = `3`

High quality of roughness filter for screen-space reflections. This is the slowest option.

enum **EnvironmentSSAOQuality**: 🔗

EnvironmentSSAOQuality **ENV_SSAO_QUALITY_VERY_LOW** = `0`

Lowest quality of screen-space ambient occlusion.

EnvironmentSSAOQuality **ENV_SSAO_QUALITY_LOW** = `1`

Low quality screen-space ambient occlusion.

EnvironmentSSAOQuality **ENV_SSAO_QUALITY_MEDIUM** = `2`

Medium quality screen-space ambient occlusion.

EnvironmentSSAOQuality **ENV_SSAO_QUALITY_HIGH** = `3`

High quality screen-space ambient occlusion.

EnvironmentSSAOQuality **ENV_SSAO_QUALITY_ULTRA** = `4`

Highest quality screen-space ambient occlusion. Uses the adaptive target setting which can be dynamically adjusted to smoothly balance performance and visual quality.

enum **EnvironmentSSILQuality**: 🔗

EnvironmentSSILQuality **ENV_SSIL_QUALITY_VERY_LOW** = `0`

Lowest quality of screen-space indirect lighting.

EnvironmentSSILQuality **ENV_SSIL_QUALITY_LOW** = `1`

Low quality screen-space indirect lighting.

EnvironmentSSILQuality **ENV_SSIL_QUALITY_MEDIUM** = `2`

High quality screen-space indirect lighting.

EnvironmentSSILQuality **ENV_SSIL_QUALITY_HIGH** = `3`

High quality screen-space indirect lighting.

EnvironmentSSILQuality **ENV_SSIL_QUALITY_ULTRA** = 4

Highest quality screen-space indirect lighting. Uses the adaptive target setting which can be dynamically adjusted to smoothly balance performance and visual quality.

enum **EnvironmentSDFGIYScale**: 🔗

EnvironmentSDFGIYScale **ENV_SDFGI_Y_SCALE_50_PERCENT** = 0

Use 50% scale for SDFGI on the Y (vertical) axis. SDFGI cells will be twice as short as they are wide. This allows providing increased GI detail and reduced light leaking with thin floors and ceilings. This is usually the best choice for scenes that don't feature much verticality.

EnvironmentSDFGIYScale **ENV_SDFGI_Y_SCALE_75_PERCENT** = 1

Use 75% scale for SDFGI on the Y (vertical) axis. This is a balance between the 50% and 100% SDFGI Y scales.

EnvironmentSDFGIYScale **ENV_SDFGI_Y_SCALE_100_PERCENT** = 2

Use 100% scale for SDFGI on the Y (vertical) axis. SDFGI cells will be as tall as they are wide. This is usually the best choice for highly vertical scenes. The downside is that light leaking may become more noticeable with thin floors and ceilings.

enum **EnvironmentSDFGIRayCount**: 🔗

EnvironmentSDFGIRayCount **ENV_SDFGI_RAY_COUNT_4** = 0

Throw 4 rays per frame when converging SDFGI. This has the lowest GPU requirements, but creates the most noisy result.

EnvironmentSDFGIRayCount **ENV_SDFGI_RAY_COUNT_8** = 1

Throw 8 rays per frame when converging SDFGI.

EnvironmentSDFGIRayCount **ENV_SDFGI_RAY_COUNT_16** = 2

Throw 16 rays per frame when converging SDFGI.

EnvironmentSDFGIRayCount **ENV_SDFGI_RAY_COUNT_32** = 3

Throw 32 rays per frame when converging SDFGI.

EnvironmentSDFGIRayCount **ENV_SDFGI_RAY_COUNT_64** = 4

Throw 64 rays per frame when converging SDFGI.

EnvironmentSDFGIRayCount **ENV_SDFGI_RAY_COUNT_96** = 5

Throw 96 rays per frame when converging SDFGI. This has high GPU requirements.

EnvironmentSDFGIRayCount **ENV_SDFGI_RAY_COUNT_128** = `6`

Throw 128 rays per frame when converging SDFGI. This has very high GPU requirements, but creates the least noisy result.

EnvironmentSDFGIRayCount **ENV_SDFGI_RAY_COUNT_MAX** = `7`

Represents the size of the EnvironmentSDFGIRayCount enum.

---

enum **EnvironmentSDFGIFramesToConverge**: 🔗

EnvironmentSDFGIFramesToConverge **ENV_SDFGI_CONVERGE_IN_5_FRAMES** = `0`

Converge SDFGI over 5 frames. This is the most responsive, but creates the most noisy result with a given ray count.

EnvironmentSDFGIFramesToConverge **ENV_SDFGI_CONVERGE_IN_10_FRAMES** = `1`

Configure SDFGI to fully converge over 10 frames.

EnvironmentSDFGIFramesToConverge **ENV_SDFGI_CONVERGE_IN_15_FRAMES** = `2`

Configure SDFGI to fully converge over 15 frames.

EnvironmentSDFGIFramesToConverge **ENV_SDFGI_CONVERGE_IN_20_FRAMES** = `3`

Configure SDFGI to fully converge over 20 frames.

EnvironmentSDFGIFramesToConverge **ENV_SDFGI_CONVERGE_IN_25_FRAMES** = `4`

Configure SDFGI to fully converge over 25 frames.

EnvironmentSDFGIFramesToConverge **ENV_SDFGI_CONVERGE_IN_30_FRAMES** = `5`

Configure SDFGI to fully converge over 30 frames. This is the least responsive, but creates the least noisy result with a given ray count.

EnvironmentSDFGIFramesToConverge **ENV_SDFGI_CONVERGE_MAX** = `6`

Represents the size of the EnvironmentSDFGIFramesToConverge enum.

---

enum **EnvironmentSDFGIFramesToUpdateLight**: 🔗

EnvironmentSDFGIFramesToUpdateLight **ENV_SDFGI_UPDATE_LIGHT_IN_1_FRAME** = `0`

Update indirect light from dynamic lights in SDFGI over 1 frame. This is the most responsive, but has the highest GPU requirements.

EnvironmentSDFGIFramesToUpdateLight **ENV_SDFGI_UPDATE_LIGHT_IN_2_FRAMES** = `1`

Update indirect light from dynamic lights in SDFGI over 2 frames.

EnvironmentSDFGIFramesToUpdateLight **ENV_SDFGI_UPDATE_LIGHT_IN_4_FRAMES** = `2`

Update indirect light from dynamic lights in SDFGI over 4 frames.

EnvironmentSDFGIFramesToUpdateLight **ENV_SDFGI_UPDATE_LIGHT_IN_8_FRAMES** = `3`

Update indirect light from dynamic lights in SDFGI over 8 frames.

EnvironmentSDFGIFramesToUpdateLight **ENV_SDFGI_UPDATE_LIGHT_IN_16_FRAMES** = `4`

Update indirect light from dynamic lights in SDFGI over 16 frames. This is the least responsive, but has the lowest GPU requirements.

EnvironmentSDFGIFramesToUpdateLight **ENV_SDFGI_UPDATE_LIGHT_MAX** = `5`

Represents the size of the EnvironmentSDFGIFramesToUpdateLight enum.

---

enum **SubSurfaceScatteringQuality**: 🔗

SubSurfaceScatteringQuality **SUB_SURFACE_SCATTERING_QUALITY_DISABLED** = `0`

Disables subsurface scattering entirely, even on materials that have BaseMaterial3D.subsurf_scatter_enabled set to `true`. This has the lowest GPU requirements.

SubSurfaceScatteringQuality **SUB_SURFACE_SCATTERING_QUALITY_LOW** = `1`

Low subsurface scattering quality.

SubSurfaceScatteringQuality **SUB_SURFACE_SCATTERING_QUALITY_MEDIUM** = `2`

Medium subsurface scattering quality.

SubSurfaceScatteringQuality **SUB_SURFACE_SCATTERING_QUALITY_HIGH** = `3`

High subsurface scattering quality. This has the highest GPU requirements.

---

enum **DOFBokehShape**: 🔗

DOFBokehShape **DOF_BOKEH_BOX** = `0`

Calculate the DOF blur using a box filter. The fastest option, but results in obvious lines in blur pattern.

DOFBokehShape **DOF_BOKEH_HEXAGON** = `1`

Calculates DOF blur using a hexagon shaped filter.

DOFBokehShape **DOF_BOKEH_CIRCLE** = `2`

Calculates DOF blur using a circle shaped filter. Best quality and most realistic, but slowest. Use only for areas where a lot of performance can be dedicated to post-processing (e.g. cutscenes).

---

enum **DOFBlurQuality**: 🔗

DOFBlurQuality **DOF_BLUR_QUALITY_VERY_LOW** = `0`

Lowest quality DOF blur. This is the fastest setting, but you may be able to see filtering artifacts.

DOFBlurQuality **DOF_BLUR_QUALITY_LOW** = `1`

Low quality DOF blur.

DOFBlurQuality **DOF_BLUR_QUALITY_MEDIUM** = `2`

Medium quality DOF blur.

DOFBlurQuality **DOF_BLUR_QUALITY_HIGH** = `3`

Highest quality DOF blur. Results in the smoothest looking blur by taking the most samples, but is also significantly slower.

---

enum **InstanceType**: 🔗

InstanceType **INSTANCE_NONE** = `0`

The instance does not have a type.

InstanceType **INSTANCE_MESH** = `1`

The instance is a mesh.

InstanceType **INSTANCE_MULTIMESH** = `2`

The instance is a multimesh.

InstanceType **INSTANCE_PARTICLES** = `3`

The instance is a particle emitter.

InstanceType **INSTANCE_PARTICLES_COLLISION** = `4`

The instance is a GPUParticles collision shape.

InstanceType **INSTANCE_LIGHT** = `5`

The instance is a light.

InstanceType **INSTANCE_REFLECTION_PROBE** = `6`

The instance is a reflection probe.

InstanceType **INSTANCE_DECAL** = `7`

The instance is a decal.

InstanceType **INSTANCE_VOXEL_GI** = `8`

The instance is a VoxelGI.

InstanceType **INSTANCE_LIGHTMAP** = `9`

The instance is a lightmap.

InstanceType **INSTANCE_OCCLUDER** = `10`

The instance is an occlusion culling occluder.

InstanceType **INSTANCE_VISIBLITY_NOTIFIER** = `11`

The instance is a visible on-screen notifier.

InstanceType **INSTANCE_FOG_VOLUME** = `12`

The instance is a fog volume.

InstanceType **INSTANCE_MAX** = `13`

Represents the size of the InstanceType enum.

InstanceType **INSTANCE_GEOMETRY_MASK** = `14`

A combination of the flags of geometry instances (mesh, multimesh, immediate and particles).

enum **InstanceFlags**:

InstanceFlags **INSTANCE_FLAG_USE_BAKED_LIGHT** = `0`

Allows the instance to be used in baked lighting.

InstanceFlags **INSTANCE_FLAG_USE_DYNAMIC_GI** = `1`

Allows the instance to be used with dynamic global illumination.

InstanceFlags **INSTANCE_FLAG_DRAW_NEXT_FRAME_IF_VISIBLE** = `2`

When set, manually requests to draw geometry on next frame.

InstanceFlags **INSTANCE_FLAG_IGNORE_OCCLUSION_CULLING** = `3`

Always draw, even if the instance would be culled by occlusion culling. Does not affect view frustum culling.

InstanceFlags **INSTANCE_FLAG_MAX** = `4`

Represents the size of the InstanceFlags enum.

enum **ShadowCastingSetting**:

ShadowCastingSetting **SHADOW_CASTING_SETTING_OFF** = `0`

Disable shadows from this instance.

ShadowCastingSetting **SHADOW_CASTING_SETTING_ON** = `1`

Cast shadows from this instance.

ShadowCastingSetting **SHADOW_CASTING_SETTING_DOUBLE_SIDED** = `2`

Disable backface culling when rendering the shadow of the object. This is slightly slower but may result in more correct shadows.

ShadowCastingSetting **SHADOW_CASTING_SETTING_SHADOWS_ONLY** = `3`

Only render the shadows from the object. The object itself will not be drawn.

enum **VisibilityRangeFadeMode**: 🔗

VisibilityRangeFadeMode **VISIBILITY_RANGE_FADE_DISABLED** = `0`

Disable visibility range fading for the given instance.

VisibilityRangeFadeMode **VISIBILITY_RANGE_FADE_SELF** = `1`

Fade-out the given instance when it approaches its visibility range limits.

VisibilityRangeFadeMode **VISIBILITY_RANGE_FADE_DEPENDENCIES** = `2`

Fade-in the given instance's dependencies when reaching its visibility range limits.

enum **BakeChannels**: 🔗

BakeChannels **BAKE_CHANNEL_ALBEDO_ALPHA** = `0`

Index of Image in array of Images returned by bake_render_uv2. Image uses Image.FORMAT_RGBA8 and contains albedo color in the `.rgb` channels and alpha in the `.a` channel.

BakeChannels **BAKE_CHANNEL_NORMAL** = `1`

Index of Image in array of Images returned by bake_render_uv2. Image uses Image.FORMAT_RGBA8 and contains the per-pixel normal of the object in the `.rgb` channels and nothing in the `.a` channel. The per-pixel normal is encoded as `normal * 0.5 + 0.5`.

BakeChannels **BAKE_CHANNEL_ORM** = `2`

Index of Image in array of Images returned by bake_render_uv2. Image uses Image.FORMAT_RGBA8 and contains ambient occlusion (from material and decals only) in the `.r` channel, roughness in the `.g` channel, metallic in the `.b` channel and sub surface scattering amount in the `.a` channel.

BakeChannels **BAKE_CHANNEL_EMISSION** = `3`

Index of Image in array of Images returned by bake_render_uv2. Image uses Image.FORMAT_RGBAH and contains emission color in the `.rgb` channels and nothing in the `.a` channel.

enum **CanvasTextureChannel**: 🔗

CanvasTextureChannel **CANVAS_TEXTURE_CHANNEL_DIFFUSE** = `0`

Diffuse canvas texture (CanvasTexture.diffuse_texture).

CanvasTextureChannel **CANVAS_TEXTURE_CHANNEL_NORMAL** = `1`

Normal map canvas texture (CanvasTexture.normal_texture).

CanvasTextureChannel **CANVAS_TEXTURE_CHANNEL_SPECULAR** = `2`

Specular map canvas texture (CanvasTexture.specular_texture).

enum **NinePatchAxisMode**: 🔗

NinePatchAxisMode **NINE_PATCH_STRETCH** = `0`

The nine patch gets stretched where needed.

NinePatchAxisMode **NINE_PATCH_TILE** = `1`

The nine patch gets filled with tiles where needed.

NinePatchAxisMode **NINE_PATCH_TILE_FIT** = `2`

The nine patch gets filled with tiles where needed and stretches them a bit if needed.

enum **CanvasItemTextureFilter**: 🔗

CanvasItemTextureFilter **CANVAS_ITEM_TEXTURE_FILTER_DEFAULT** = `0`

Uses the default filter mode for this Viewport.

CanvasItemTextureFilter **CANVAS_ITEM_TEXTURE_FILTER_NEAREST** = `1`

The texture filter reads from the nearest pixel only. This makes the texture look pixelated from up close, and grainy from a distance (due to mipmaps not being sampled).

CanvasItemTextureFilter **CANVAS_ITEM_TEXTURE_FILTER_LINEAR** = `2`

The texture filter blends between the nearest 4 pixels. This makes the texture look smooth from up close, and grainy from a distance (due to mipmaps not being sampled).

CanvasItemTextureFilter **CANVAS_ITEM_TEXTURE_FILTER_NEAREST_WITH_MIPMAPS** = `3`

The texture filter reads from the nearest pixel and blends between the nearest 2 mipmaps (or uses the nearest mipmap if ProjectSettings.rendering/textures/default_filters/use_nearest_mipmap_filter is `true`). This makes the texture look pixelated from up close, and smooth from a distance.

Use this for non-pixel art textures that may be viewed at a low scale (e.g. due to Camera2D zoom or sprite scaling), as mipmaps are important to smooth out pixels that are smaller than on-screen pixels.

CanvasItemTextureFilter **CANVAS_ITEM_TEXTURE_FILTER_LINEAR_WITH_MIPMAPS** = `4`

The texture filter blends between the nearest 4 pixels and between the nearest 2 mipmaps (or uses the nearest mipmap if ProjectSettings.rendering/textures/default_filters/use_nearest_mipmap_filter is `true`). This makes the texture look smooth from up close, and smooth from a distance.

Use this for non-pixel art textures that may be viewed at a low scale (e.g. due to Camera2D zoom or sprite scaling), as mipmaps are important to smooth out pixels that are smaller than on-screen pixels.

CanvasItemTextureFilter **CANVAS_ITEM_TEXTURE_FILTER_NEAREST_WITH_MIPMAPS_ANISOTROPIC** = `5`

The texture filter reads from the nearest pixel and blends between 2 mipmaps (or uses the nearest mipmap if ProjectSettings.rendering/textures/default_filters/use_nearest_mipmap_filter is `true`) based on the angle between the surface and the camera view. This makes the texture look pixelated from up close, and smooth from a distance. Anisotropic filtering improves texture quality on surfaces that are almost in line with the camera, but is slightly slower. The anisotropic filtering level can be changed by adjusting ProjectSettings.rendering/textures/default_filters/anisotropic_filtering_level.

Note: This texture filter is rarely useful in 2D projects. CANVAS_ITEM_TEXTURE_FILTER_NEAREST_WITH_MIPMAPS is usually more appropriate in this case.

CanvasItemTextureFilter **CANVAS_ITEM_TEXTURE_FILTER_LINEAR_WITH_MIPMAPS_ANISOTROPIC** = `6`

The texture filter blends between the nearest 4 pixels and blends between 2 mipmaps (or uses the nearest mipmap if ProjectSettings.rendering/textures/default_filters/use_nearest_mipmap_filter is `true`) based on the angle between the surface and the camera view. This makes the texture look smooth from up close, and smooth from a distance. Anisotropic filtering improves texture quality on surfaces that are almost in line with the camera, but is slightly slower. The anisotropic filtering level can be changed by adjusting ProjectSettings.rendering/textures/default_filters/anisotropic_filtering_level.

Note: This texture filter is rarely useful in 2D projects. CANVAS_ITEM_TEXTURE_FILTER_LINEAR_WITH_MIPMAPS is usually more appropriate in this case.

CanvasItemTextureFilter **CANVAS_ITEM_TEXTURE_FILTER_MAX** = `7`

Max value for CanvasItemTextureFilter enum.

---

enum **CanvasItemTextureRepeat**: 🔗

CanvasItemTextureRepeat **CANVAS_ITEM_TEXTURE_REPEAT_DEFAULT** = `0`

Uses the default repeat mode for this Viewport.

CanvasItemTextureRepeat **CANVAS_ITEM_TEXTURE_REPEAT_DISABLED** = `1`

Disables textures repeating. Instead, when reading UVs outside the 0-1 range, the value will be clamped to the edge of the texture, resulting in a stretched out look at the borders of the texture.

CanvasItemTextureRepeat **CANVAS_ITEM_TEXTURE_REPEAT_ENABLED** = `2`

Enables the texture to repeat when UV coordinates are outside the 0-1 range. If using one of the linear filtering modes, this can result in artifacts at the edges of a texture when the sampler filters across the edges of the texture.

CanvasItemTextureRepeat **CANVAS_ITEM_TEXTURE_REPEAT_MIRROR** = `3`

Flip the texture when repeating so that the edge lines up instead of abruptly changing.

CanvasItemTextureRepeat **CANVAS_ITEM_TEXTURE_REPEAT_MAX** = `4`

Max value for CanvasItemTextureRepeat enum.

---

enum **CanvasGroupMode**: 🔗

CanvasGroupMode **CANVAS_GROUP_MODE_DISABLED** = `0`

Child draws over parent and is not clipped.

CanvasGroupMode **CANVAS_GROUP_MODE_CLIP_ONLY** = `1`

Parent is used for the purposes of clipping only. Child is clipped to the parent's visible area, parent is not drawn.

CanvasGroupMode **CANVAS_GROUP_MODE_CLIP_AND_DRAW** = `2`

Parent is used for clipping child, but parent is also drawn underneath child as normal before clipping child to its visible area.

CanvasGroupMode **CANVAS_GROUP_MODE_TRANSPARENT** = `3`

There is currently no description for this enum. Please help us by contributing one!

---

enum **CanvasLightMode**: 🔗

CanvasLightMode **CANVAS_LIGHT_MODE_POINT** = `0`

2D point light (see PointLight2D).

CanvasLightMode **CANVAS_LIGHT_MODE_DIRECTIONAL** = `1`

2D directional (sun/moon) light (see DirectionalLight2D).

---

enum **CanvasLightBlendMode**: 🔗

CanvasLightBlendMode **CANVAS_LIGHT_BLEND_MODE_ADD** = `0`

Adds light color additive to the canvas.

CanvasLightBlendMode **CANVAS_LIGHT_BLEND_MODE_SUB** = `1`

Adds light color subtractive to the canvas.

CanvasLightBlendMode **CANVAS_LIGHT_BLEND_MODE_MIX** = `2`

The light adds color depending on transparency.

---

enum **CanvasLightShadowFilter**: 🔗

CanvasLightShadowFilter **CANVAS_LIGHT_FILTER_NONE** = `0`

Do not apply a filter to canvas light shadows.

CanvasLightShadowFilter **CANVAS_LIGHT_FILTER_PCF5** = `1`

Use PCF5 filtering to filter canvas light shadows.

CanvasLightShadowFilter **CANVAS_LIGHT_FILTER_PCF13** = `2`

Use PCF13 filtering to filter canvas light shadows.

CanvasLightShadowFilter **CANVAS_LIGHT_FILTER_MAX** = `3`

Max value of the CanvasLightShadowFilter enum.

---

enum **CanvasOccluderPolygonCullMode**: 🔗

CanvasOccluderPolygonCullMode **CANVAS_OCCLUDER_POLYGON_CULL_DISABLED** = `0`

Culling of the canvas occluder is disabled.

CanvasOccluderPolygonCullMode **CANVAS_OCCLUDER_POLYGON_CULL_CLOCKWISE** = `1`

Culling of the canvas occluder is clockwise.

CanvasOccluderPolygonCullMode
CANVAS_OCCLUDER_POLYGON_CULL_COUNTER_CLOCKWISE = `2`

Culling of the canvas occluder is counterclockwise.

---

enum **GlobalShaderParameterType**: 🔗

GlobalShaderParameterType **GLOBAL_VAR_TYPE_BOOL** = `0`

Boolean global shader parameter (`global uniform bool ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_BVEC2** = `1`

2-dimensional boolean vector global shader parameter (`global uniform bvec2 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_BVEC3** = `2`

3-dimensional boolean vector global shader parameter (`global uniform bvec3 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_BVEC4** = `3`

4-dimensional boolean vector global shader parameter (`global uniform bvec4 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_INT** = `4`

Integer global shader parameter (`global uniform int ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_IVEC2** = `5`

2-dimensional integer vector global shader parameter (`global uniform ivec2 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_IVEC3** = `6`

3-dimensional integer vector global shader parameter (`global uniform ivec3 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_IVEC4** = `7`

4-dimensional integer vector global shader parameter (`global uniform ivec4 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_RECT2I** = `8`

2-dimensional integer rectangle global shader parameter (`global uniform ivec4 ...`). Equivalent to GLOBAL_VAR_TYPE_IVEC4 in shader code, but exposed as a Rect2i in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_UINT** = `9`

Unsigned integer global shader parameter (`global uniform uint ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_UVEC2** = `10`

2-dimensional unsigned integer vector global shader parameter (`global uniform uvec2 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_UVEC3** = `11`

3-dimensional unsigned integer vector global shader parameter (`global uniform uvec3 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_UVEC4** = `12`

4-dimensional unsigned integer vector global shader parameter (`global uniform uvec4 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_FLOAT** = `13`

Single-precision floating-point global shader parameter (`global uniform float ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_VEC2** = `14`

2-dimensional floating-point vector global shader parameter (`global uniform vec2 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_VEC3** = `15`

3-dimensional floating-point vector global shader parameter (`global uniform vec3 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_VEC4** = `16`

4-dimensional floating-point vector global shader parameter (`global uniform vec4 ...`).

GlobalShaderParameterType **GLOBAL_VAR_TYPE_COLOR** = `17`

Color global shader parameter (`global uniform vec4 ...`). Equivalent to GLOBAL_VAR_TYPE_VEC4 in shader code, but exposed as a Color in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_RECT2** = `18`

2-dimensional floating-point rectangle global shader parameter (`global uniform vec4 ...`). Equivalent to GLOBAL_VAR_TYPE_VEC4 in shader code, but exposed as a Rect2 in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_MAT2** = `19`

2×2 matrix global shader parameter (`global uniform mat2 ...`). Exposed as a PackedInt32Array in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_MAT3** = `20`

3×3 matrix global shader parameter (`global uniform mat3 ...`). Exposed as a Basis in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_MAT4** = `21`

4×4 matrix global shader parameter (`global uniform mat4 ...`). Exposed as a Projection in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_TRANSFORM_2D** = `22`

2-dimensional transform global shader parameter (`global uniform mat2x3 ...`). Exposed as a Transform2D in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_TRANSFORM** = `23`

3-dimensional transform global shader parameter (`global uniform mat3x4 ...`). Exposed as a Transform3D in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_SAMPLER2D** = `24`

2D sampler global shader parameter (`global uniform sampler2D ...`). Exposed as a Texture2D in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_SAMPLER2DARRAY** = `25`

2D sampler array global shader parameter (`global uniform sampler2DArray ...`). Exposed as a Texture2DArray in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_SAMPLER3D** = `26`

3D sampler global shader parameter (`global uniform sampler3D ...`). Exposed as a Texture3D in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_SAMPLERCUBE** = `27`

Cubemap sampler global shader parameter (`global uniform samplerCube ...`). Exposed as a Cubemap in the editor UI.

GlobalShaderParameterType **GLOBAL_VAR_TYPE_MAX** = `28`

Represents the size of the GlobalShaderParameterType enum.

---

enum **RenderingInfo**: 🔗

RenderingInfo **RENDERING_INFO_TOTAL_OBJECTS_IN_FRAME** = `0`

Number of objects rendered in the current 3D scene. This varies depending on camera position and rotation.

RenderingInfo **RENDERING_INFO_TOTAL_PRIMITIVES_IN_FRAME** = `1`

Number of points, lines, or triangles rendered in the current 3D scene. This varies depending on camera position and rotation.

RenderingInfo **RENDERING_INFO_TOTAL_DRAW_CALLS_IN_FRAME** = `2`

Number of draw calls performed to render in the current 3D scene. This varies depending on camera position and rotation.

RenderingInfo **RENDERING_INFO_TEXTURE_MEM_USED** = `3`

Texture memory used (in bytes).

RenderingInfo **RENDERING_INFO_BUFFER_MEM_USED** = `4`

Buffer memory used (in bytes). This includes vertex data, uniform buffers, and many miscellaneous buffer types used internally.

RenderingInfo **RENDERING_INFO_VIDEO_MEM_USED** = `5`

Video memory used (in bytes). When using the Forward+ or mobile rendering backends, this is always greater than the sum of RENDERING_INFO_TEXTURE_MEM_USED and RENDERING_INFO_BUFFER_MEM_USED, since there is miscellaneous data not accounted for by those two metrics. When using the GL Compatibility backend, this is equal to the sum of RENDERING_INFO_TEXTURE_MEM_USED and RENDERING_INFO_BUFFER_MEM_USED.

---

enum **Features**: 🔗

Features **FEATURE_SHADERS** = `0`

Deprecated: This constant has not been used since Godot 3.0.

Features **FEATURE_MULTITHREADED** = `1`

Deprecated: This constant has not been used since Godot 3.0.

Constants

NO_INDEX_ARRAY = `-1` 🔗

Marks an error that shows that the index array is empty.

ARRAY_WEIGHTS_SIZE = `4` 🔗

Number of weights/bones per vertex.

CANVAS_ITEM_Z_MIN = `-4096` 🔗

The minimum Z-layer for canvas items.

CANVAS_ITEM_Z_MAX = `4096` 🔗

The maximum Z-layer for canvas items.

MAX_GLOW_LEVELS = `7` 🔗

The maximum number of glow levels that can be used with the glow post-processing effect.

MAX_CURSORS = `8` 🔗

Deprecated: This constant is not used by the engine.

MAX_2D_DIRECTIONAL_LIGHTS = `8` 🔗

The maximum number of directional lights that can be rendered at a given time in 2D.

MAX_MESH_SURFACES = `256` 🔗

The maximum number of surfaces a mesh can have.

MATERIAL_RENDER_PRIORITY_MIN = `-128` 🔗

The minimum renderpriority of all materials.

MATERIAL_RENDER_PRIORITY_MAX = `127` 🔗

The maximum renderpriority of all materials.

ARRAY_CUSTOM_COUNT = `4` 🔗

The number of custom data arrays available (ARRAY_CUSTOM0, ARRAY_CUSTOM1, ARRAY_CUSTOM2, ARRAY_CUSTOM3).

PARTICLES_EMIT_FLAG_POSITION = `1` 🔗

There is currently no description for this constant. Please help us by contributing one!

PARTICLES_EMIT_FLAG_ROTATION_SCALE = `2` 🔗

There is currently no description for this constant. Please help us by contributing one!

PARTICLES_EMIT_FLAG_VELOCITY = `4`

There is currently no description for this constant. Please help us by contributing one!

PARTICLES_EMIT_FLAG_COLOR = `8`

There is currently no description for this constant. Please help us by contributing one!

PARTICLES_EMIT_FLAG_CUSTOM = `16`

There is currently no description for this constant. Please help us by contributing one!

Property Descriptions

bool **render_loop_enabled**

- void **set_render_loop_enabled**(value: bool)
- bool **is_render_loop_enabled**()

If `false`, disables rendering completely, but the engine logic is still being processed. You can call force_draw to draw a frame even with rendering disabled.

Method Descriptions

Array[Image] **bake_render_uv2**(base: RID, material_overrides: Array[RID], image_size: Vector2i)

Bakes the material data of the Mesh passed in the `base` parameter with optional `material_overrides` to a set of Images of size `image_size`. Returns an array of Images containing material properties as specified in BakeChannels.

---

void **call_on_render_thread**(callable: Callable)

As the RenderingServer actual logic may run on an separate thread, accessing its internals from the main (or any other) thread will result in errors. To make it easier to run code that can safely access the rendering internals (such as RenderingDevice and similar RD classes), push a callable via this function so it will be executed on the render thread.

RID **camera_attributes_create**()

Creates a camera attributes object and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `camera_attributes_` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is CameraAttributes.

---

void **camera_attributes_set_auto_exposure**(camera_attributes: RID, enable: bool, min_sensitivity: float, max_sensitivity: float, speed: float, scale: float)

Sets the parameters to use with the auto-exposure effect. These parameters take on the same meaning as their counterparts in CameraAttributes and CameraAttributesPractical.

---

void **camera_attributes_set_dof_blur**(camera_attributes: RID, far_enable: bool, far_distance: float, far_transition: float, near_enable: bool, near_distance: float, near_transition: float, amount: float)

Sets the parameters to use with the DOF blur effect. These parameters take on the same meaning as their counterparts in CameraAttributesPractical.

---

void **camera_attributes_set_dof_blur_bokeh_shape**(shape: DOFBokehShape)

Sets the shape of the DOF bokeh pattern. Different shapes may be used to achieve artistic effect, or to meet performance targets. For more detail on available options see DOFBokehShape.

---

void **camera_attributes_set_dof_blur_quality**(quality: DOFBlurQuality, use_jitter: bool)

Sets the quality level of the DOF blur effect to one of the options in DOFBlurQuality. `use_jitter` can be used to jitter samples taken during the blur pass to hide artifacts at the cost of looking more fuzzy.

---

void **camera_attributes_set_exposure**(camera_attributes: RID, multiplier: float, normalization: float)

Sets the exposure values that will be used by the renderers. The normalization amount is used to bake a given Exposure Value (EV) into rendering calculations to reduce the dynamic range of the scene.

The normalization factor can be calculated from exposure value (EV100) as follows:

```
func get_exposure_normalization(ev100: float):
    return 1.0 / (pow(2.0, ev100) * 1.2)
```

The exposure value can be calculated from aperture (in f-stops), shutter speed (in seconds), and sensitivity (in ISO) as follows:

```
func get_exposure(aperture: float, shutter_speed: float, sensitivity: float):
    return log((aperture * aperture) / shutter_speed * (100.0 / sensitivity)) / log(2)
```

RID **camera_create**()

Creates a 3D camera and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `camera_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent node is Camera3D.

void **camera_set_camera_attributes**(camera: RID, effects: RID)

Sets the camera_attributes created with camera_attributes_create to the given camera.

void **camera_set_compositor**(camera: RID, compositor: RID)

Sets the compositor used by this camera. Equivalent to Camera3D.compositor.

void **camera_set_cull_mask**(camera: RID, layers: int)

Sets the cull mask associated with this camera. The cull mask describes which 3D layers are rendered by this camera. Equivalent to Camera3D.cull_mask.

void **camera_set_environment**(camera: RID, env: RID)

Sets the environment used by this camera. Equivalent to Camera3D.environment.

void **camera_set_frustum**(camera: RID, size: float, offset: Vector2, z_near: float, z_far: float)

Sets camera to use frustum projection. This mode allows adjusting the `offset` argument to create "tilted frustum" effects.

void **camera_set_orthogonal**(camera: RID, size: float, z_near: float, z_far: float)

Sets camera to use orthogonal projection, also known as orthographic projection. Objects remain the same size on the screen no matter how far away they are.

void **camera_set_perspective**(camera: RID, fovy_degrees: float, z_near: float, z_far: float)

Sets camera to use perspective projection. Objects on the screen becomes smaller when they are far away.

void **camera_set_transform**(camera: RID, transform: Transform3D)

Sets Transform3D of camera.

void **camera_set_use_vertical_aspect**(camera: RID, enable: bool)

If `true`, preserves the horizontal aspect ratio which is equivalent to Camera3D.KEEP_WIDTH. If `false`, preserves the vertical aspect ratio which is equivalent to Camera3D.KEEP_HEIGHT.

RID **canvas_create**()

Creates a canvas and returns the assigned RID. It can be accessed with the RID that is returned. This RID will be used in all `canvas_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Canvas has no Resource or Node equivalent.

void **canvas_item_add_animation_slice**(item: RID, animation_length: float, slice_begin: float, slice_end: float, offset: float = 0.0)

Subsequent drawing commands will be ignored unless they fall within the specified animation slice. This is a faster way to implement animations that loop on background rather than redrawing constantly.

void **canvas_item_add_circle**(item: RID, pos: Vector2, radius: float, color: Color, antialiased: bool = false)

Draws a circle on the CanvasItem pointed to by the `item` RID. See also CanvasItem.draw_circle.

void **canvas_item_add_clip_ignore**(item: RID, ignore: bool)

If `ignore` is `true`, ignore clipping on items drawn with this canvas item until this is called again with `ignore` set to false.

void **canvas_item_add_lcd_texture_rect_region**(item: RID, rect: Rect2, texture: RID, src_rect: Rect2, modulate: Color)

See also CanvasItem.draw_lcd_texture_rect_region.

void **canvas_item_add_line**(item: RID, from: Vector2, to: Vector2, color: Color, width: float = -1.0, antialiased: bool = false)

Draws a line on the CanvasItem pointed to by the `item` RID. See also CanvasItem.draw_line.

void **canvas_item_add_mesh**(item: RID, mesh: RID, transform: Transform2D = Transform2D(1, 0, 0, 1, 0, 0), modulate: Color = Color(1, 1, 1, 1), texture: RID = RID())

Draws a mesh created with mesh_create with given `transform`, `modulate` color, and `texture`. This is used internally by MeshInstance2D.

void **canvas_item_add_msdf_texture_rect_region**(item: RID, rect: Rect2, texture: RID, src_rect: Rect2, modulate: Color = Color(1, 1, 1, 1), outline_size: int = 0, px_range: float = 1.0, scale: float = 1.0)

See also CanvasItem.draw_msdf_texture_rect_region.

void **canvas_item_add_multiline**(item: RID, points: PackedVector2Array, colors: PackedColorArray, width: float = -1.0, antialiased: bool = false)

Draws a 2D multiline on the CanvasItem pointed to by the `item` RID. See also CanvasItem.draw_multiline and CanvasItem.draw_multiline_colors.

void **canvas_item_add_multimesh**(item: RID, mesh: RID, texture: RID = RID())

Draws a 2D MultiMesh on the CanvasItem pointed to by the `item` RID. See also CanvasItem.draw_multimesh.

void **canvas_item_add_nine_patch**(item: RID, rect: Rect2, source: Rect2, texture: RID, topleft: Vector2, bottomright: Vector2, x_axis_mode: NinePatchAxisMode = 0, y_axis_mode: NinePatchAxisMode = 0, draw_center: bool = true, modulate: Color = Color(1, 1, 1, 1))

Draws a nine-patch rectangle on the CanvasItem pointed to by the `item` RID.

void **canvas_item_add_particles**(item: RID, particles: RID, texture: RID)

Draws particles on the CanvasItem pointed to by the `item` RID.

void **canvas_item_add_polygon**(item: RID, points: PackedVector2Array, colors: PackedColorArray, uvs: PackedVector2Array = PackedVector2Array(), texture: RID = RID())

Draws a 2D polygon on the CanvasItem pointed to by the `item` RID. If you need more flexibility (such as being able to use bones), use canvas_item_add_triangle_array instead. See also CanvasItem.draw_polygon.

void **canvas_item_add_polyline**(item: RID, points: PackedVector2Array, colors: PackedColorArray, width: float = -1.0, antialiased: bool = false)

Draws a 2D polyline on the CanvasItem pointed to by the `item` RID. See also CanvasItem.draw_polyline and CanvasItem.draw_polyline_colors.

void **canvas_item_add_primitive**(item: RID, points: PackedVector2Array, colors: PackedColorArray, uvs: PackedVector2Array, texture: RID)

Draws a 2D primitive on the CanvasItem pointed to by the `item` RID. See also CanvasItem.draw_primitive.

void **canvas_item_add_rect**(item: RID, rect: Rect2, color: Color, antialiased: bool = false)

Draws a rectangle on the CanvasItem pointed to by the `item` RID. See also CanvasItem.draw_rect.

void **canvas_item_add_set_transform**(item: RID, transform: Transform2D) 🔗

Sets a Transform2D that will be used to transform subsequent canvas item commands.

---

void **canvas_item_add_texture_rect**(item: RID, rect: Rect2, texture: RID, tile: bool = false, modulate: Color = Color(1, 1, 1, 1), transpose: bool = false) 🔗

Draws a 2D textured rectangle on the CanvasItem pointed to by the `item` RID. See also CanvasItem.draw_texture_rect and Texture2D.draw_rect.

---

void **canvas_item_add_texture_rect_region**(item: RID, rect: Rect2, texture: RID, src_rect: Rect2, modulate: Color = Color(1, 1, 1, 1), transpose: bool = false, clip_uv: bool = true) 🔗

Draws the specified region of a 2D textured rectangle on the CanvasItem pointed to by the `item` RID. See also CanvasItem.draw_texture_rect_region and Texture2D.draw_rect_region.

---

void **canvas_item_add_triangle_array**(item: RID, indices: PackedInt32Array, points: PackedVector2Array, colors: PackedColorArray, uvs: PackedVector2Array = PackedVector2Array(), bones: PackedInt32Array = PackedInt32Array(), weights: PackedFloat32Array = PackedFloat32Array(), texture: RID = RID(), count: int = -1) 🔗

Draws a triangle array on the CanvasItem pointed to by the `item` RID. This is internally used by Line2D and StyleBoxFlat for rendering. canvas_item_add_triangle_array is highly flexible, but more complex to use than canvas_item_add_polygon.

Note: `count` is unused and can be left unspecified.

---

void **canvas_item_clear**(item: RID) 🔗

Clears the CanvasItem and removes all commands in it.

RID **canvas_item_create**() 🔗

Creates a new CanvasItem instance and returns its RID. It can be accessed with the RID that is returned. This RID will be used in all `canvas_item_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent node is CanvasItem.

---

void **canvas_item_reset_physics_interpolation**(item: RID) 🔗

Prevents physics interpolation for the current physics tick.

This is useful when moving a canvas item to a new location, to give an instantaneous change rather than interpolation from the previous location.

---

void **canvas_item_set_canvas_group_mode**(item: RID, mode: CanvasGroupMode, clear_margin: float = 5.0, fit_empty: bool = false, fit_margin: float = 0.0, blur_mipmaps: bool = false) 🔗

Sets the canvas group mode used during 2D rendering for the canvas item specified by the `item` RID. For faster but more limited clipping, use canvas_item_set_clip instead.

Note: The equivalent node functionality is found in CanvasGroup and CanvasItem.clip_children.

---

void **canvas_item_set_clip**(item: RID, clip: bool) 🔗

If `clip` is `true`, makes the canvas item specified by the `item` RID not draw anything outside of its rect's coordinates. This clipping is fast, but works only with axis-aligned rectangles. This means that rotation is ignored by the clipping rectangle. For more advanced clipping shapes, use canvas_item_set_canvas_group_mode instead.

Note: The equivalent node functionality is found in Label.clip_text, RichTextLabel (always enabled) and more.

---

void **canvas_item_set_copy_to_backbuffer**(item: RID, enabled: bool, rect: Rect2) 🔗

Sets the CanvasItem to copy a rect to the backbuffer.

---

void **canvas_item_set_custom_rect**(item: RID, use_custom_rect: bool, rect: Rect2 = Rect2(0, 0, 0, 0)) 🔗

If `use_custom_rect` is `true`, sets the custom visibility rectangle (used for culling) to `rect` for the canvas item specified by `item`. Setting a custom visibility rect can reduce CPU load when drawing lots of 2D instances. If `use_custom_rect` is `false`, automatically computes a visibility rectangle based on the canvas item's draw commands.

---

void **canvas_item_set_default_texture_filter**(item: RID, filter: CanvasItemTextureFilter) 🔗

Sets the default texture filter mode for the canvas item specified by the `item` RID. Equivalent to CanvasItem.texture_filter.

---

void **canvas_item_set_default_texture_repeat**(item: RID, repeat: CanvasItemTextureRepeat) 🔗

Sets the default texture repeat mode for the canvas item specified by the `item` RID. Equivalent to CanvasItem.texture_repeat.

---

void **canvas_item_set_distance_field_mode**(item: RID, enabled: bool) 🔗

If `enabled` is `true`, enables multichannel signed distance field rendering mode for the canvas item specified by the `item` RID. This is meant to be used for font rendering, or with specially generated images using msdfgen [https://github.com/Chlumsky/msdfgen].

void **canvas_item_set_draw_behind_parent**(item: RID, enabled: bool)

If `enabled` is `true`, draws the canvas item specified by the `item` RID behind its parent. Equivalent to CanvasItem.show_behind_parent.

void **canvas_item_set_draw_index**(item: RID, index: int)

Sets the index for the CanvasItem.

void **canvas_item_set_interpolated**(item: RID, interpolated: bool)

If `interpolated` is `true`, turns on physics interpolation for the canvas item.

void **canvas_item_set_light_mask**(item: RID, mask: int)

Sets the light `mask` for the canvas item specified by the `item` RID. Equivalent to CanvasItem.light_mask.

void **canvas_item_set_material**(item: RID, material: RID)

Sets a new `material` to the canvas item specified by the `item` RID. Equivalent to CanvasItem.material.

void **canvas_item_set_modulate**(item: RID, color: Color)

Multiplies the color of the canvas item specified by the `item` RID, while affecting its children. See also canvas_item_set_self_modulate. Equivalent to CanvasItem.modulate.

void **canvas_item_set_parent**(item: RID, parent: RID)

Sets a parent CanvasItem to the CanvasItem. The item will inherit transform, modulation and visibility from its parent, like CanvasItem nodes in the scene tree.

void **canvas_item_set_self_modulate**(item: RID, color: Color)

Multiplies the color of the canvas item specified by the `item` RID, without affecting its children. See also canvas_item_set_modulate. Equivalent to CanvasItem.self_modulate.

void **canvas_item_set_sort_children_by_y**(item: RID, enabled: bool)

If `enabled` is `true`, child nodes with the lowest Y position are drawn before those with a higher Y position. Y-sorting only affects children that inherit from the canvas item specified by the `item` RID, not the canvas item itself. Equivalent to CanvasItem.y_sort_enabled.

void **canvas_item_set_transform**(item: RID, transform: Transform2D) 🔗

Sets the `transform` of the canvas item specified by the `item` RID. This affects where and how the item will be drawn. Child canvas items' transforms are multiplied by their parent's transform. Equivalent to Node2D.transform.

---

void **canvas_item_set_use_parent_material**(item: RID, enabled: bool) 🔗

Sets if the CanvasItem uses its parent's material.

---

void **canvas_item_set_visibility_layer**(item: RID, visibility_layer: int) 🔗

Sets the rendering visibility layer associated with this CanvasItem. Only Viewport nodes with a matching rendering mask will render this CanvasItem.

---

void **canvas_item_set_visibility_notifier**(item: RID, enable: bool, area: Rect2, enter_callable: Callable, exit_callable: Callable) 🔗

Sets the given CanvasItem as visibility notifier. `area` defines the area of detecting visibility. `enter_callable` is called when the CanvasItem enters the screen, `exit_callable` is called when the CanvasItem exits the screen. If `enable` is `false`, the item will no longer function as notifier.

This method can be used to manually mimic VisibleOnScreenNotifier2D.

---

void **canvas_item_set_visible**(item: RID, visible: bool) 🔗

Sets the visibility of the CanvasItem.

---

void **canvas_item_set_z_as_relative_to_parent**(item: RID, enabled: bool) 🔗

If this is enabled, the Z index of the parent will be added to the children's Z index.

---

void **canvas_item_set_z_index**(item: RID, z_index: int) 🔗

Sets the CanvasItem's Z index, i.e. its draw order (lower indexes are drawn first).

---

void **canvas_item_transform_physics_interpolation**(item: RID, transform: Transform2D) 🔗

Transforms both the current and previous stored transform for a canvas item.

This allows transforming a canvas item without creating a "glitch" in the interpolation, which is particularly useful for large worlds utilizing a shifting origin.

---

void **canvas_light_attach_to_canvas**(light: RID, canvas: RID) 🔗

Attaches the canvas light to the canvas. Removes it from its previous canvas.

RID **canvas_light_create**()

Creates a canvas light and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `canvas_light_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent node is Light2D.

---

void **canvas_light_occluder_attach_to_canvas**(occluder: RID, canvas: RID)

Attaches a light occluder to the canvas. Removes it from its previous canvas.

RID **canvas_light_occluder_create**()

Creates a light occluder and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `canvas_light_occluder_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent node is LightOccluder2D.

---

void **canvas_light_occluder_reset_physics_interpolation**(occluder: RID)

Prevents physics interpolation for the current physics tick.

This is useful when moving an occluder to a new location, to give an instantaneous change rather than interpolation from the previous location.

---

void **canvas_light_occluder_set_as_sdf_collision**(occluder: RID, enable: bool)

There is currently no description for this method. Please help us by contributing one!

---

void **canvas_light_occluder_set_enabled**(occluder: RID, enabled: bool)

Enables or disables light occluder.

---

void **canvas_light_occluder_set_interpolated**(occluder: RID, interpolated: bool)

If `interpolated` is `true`, turns on physics interpolation for the light occluder.

---

void **canvas_light_occluder_set_light_mask**(occluder: RID, mask: int)

The light mask. See LightOccluder2D for more information on light masks.

void **canvas_light_occluder_set_polygon**(occluder: RID, polygon: RID)

Sets a light occluder's polygon.

void **canvas_light_occluder_set_transform**(occluder: RID, transform: Transform2D)

Sets a light occluder's Transform2D.

void **canvas_light_occluder_transform_physics_interpolation**(occluder: RID, transform: Transform2D)

Transforms both the current and previous stored transform for a light occluder.

This allows transforming an occluder without creating a "glitch" in the interpolation, which is particularly useful for large worlds utilizing a shifting origin.

void **canvas_light_reset_physics_interpolation**(light: RID)

Prevents physics interpolation for the current physics tick.

This is useful when moving a canvas item to a new location, to give an instantaneous change rather than interpolation from the previous location.

void **canvas_light_set_blend_mode**(light: RID, mode: CanvasLightBlendMode)

Sets the blend mode for the given canvas light. See CanvasLightBlendMode for options. Equivalent to Light2D.blend_mode.

void **canvas_light_set_color**(light: RID, color: Color)

Sets the color for a light.

void **canvas_light_set_enabled**(light: RID, enabled: bool)

Enables or disables a canvas light.

void **canvas_light_set_energy**(light: RID, energy: float)

Sets a canvas light's energy.

void **canvas_light_set_height**(light: RID, height: float)

Sets a canvas light's height.

void **canvas_light_set_interpolated**(light: RID, interpolated: bool)

If `interpolated` is `true`, turns on physics interpolation for the canvas light.

void **canvas_light_set_item_cull_mask**(light: RID, mask: int)

The light mask. See LightOccluder2D for more information on light masks.

void **canvas_light_set_item_shadow_cull_mask**(light: RID, mask: int)

The binary mask used to determine which layers this canvas light's shadows affects. See LightOccluder2D for more information on light masks.

void **canvas_light_set_layer_range**(light: RID, min_layer: int, max_layer: int)

The layer range that gets rendered with this light.

void **canvas_light_set_mode**(light: RID, mode: CanvasLightMode)

The mode of the light, see CanvasLightMode constants.

void **canvas_light_set_shadow_color**(light: RID, color: Color)

Sets the color of the canvas light's shadow.

void **canvas_light_set_shadow_enabled**(light: RID, enabled: bool)

Enables or disables the canvas light's shadow.

void **canvas_light_set_shadow_filter**(light: RID, filter: CanvasLightShadowFilter)

Sets the canvas light's shadow's filter, see CanvasLightShadowFilter constants.

void **canvas_light_set_shadow_smooth**(light: RID, smooth: float)

Smoothens the shadow. The lower, the smoother.

void **canvas_light_set_texture**(light: RID, texture: RID)

Sets the texture to be used by a PointLight2D. Equivalent to PointLight2D.texture.

void **canvas_light_set_texture_offset**(light: RID, offset: Vector2)

Sets the offset of a PointLight2D's texture. Equivalent to PointLight2D.offset.

---

void **canvas_light_set_texture_scale**(light: RID, scale: float) 🔗

Sets the scale factor of a PointLight2D's texture. Equivalent to PointLight2D.texture_scale.

---

void **canvas_light_set_transform**(light: RID, transform: Transform2D) 🔗

Sets the canvas light's Transform2D.

---

void **canvas_light_set_z_range**(light: RID, min_z: int, max_z: int) 🔗

Sets the Z range of objects that will be affected by this light. Equivalent to Light2D.range_z_min and Light2D.range_z_max.

---

void **canvas_light_transform_physics_interpolation**(light: RID, transform: Transform2D) 🔗

Transforms both the current and previous stored transform for a canvas light.

This allows transforming a light without creating a "glitch" in the interpolation, which is is particularly useful for large worlds utilizing a shifting origin.

RID **canvas_occluder_polygon_create**() 🔗

Creates a new light occluder polygon and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `canvas_occluder_polygon_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is OccluderPolygon2D.

---

void **canvas_occluder_polygon_set_cull_mode**(occluder_polygon: RID, mode: CanvasOccluderPolygonCullMode) 🔗

Sets an occluder polygons cull mode. See CanvasOccluderPolygonCullMode constants.

---

void **canvas_occluder_polygon_set_shape**(occluder_polygon: RID, shape: PackedVector2Array, closed: bool) 🔗

Sets the shape of the occluder polygon.

---

void **canvas_set_disable_scale**(disable: bool) 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **canvas_set_item_mirroring**(canvas: RID, item: RID, mirroring: Vector2) 🔗

A copy of the canvas item will be drawn with a local offset of the `mirroring`.

Note: This is equivalent to calling canvas_set_item_repeat like `canvas_set_item_repeat(item, mirroring, 1)`, with an additional check ensuring `canvas` is a parent of `item`.

void **canvas_set_item_repeat**(item: RID, repeat_size: Vector2, repeat_times: int) 🔗

A copy of the canvas item will be drawn with a local offset of the `repeat_size` by the number of times of the `repeat_times`. As the `repeat_times` increases, the copies will spread away from the origin texture.

void **canvas_set_modulate**(canvas: RID, color: Color) 🔗

Modulates all colors in the given canvas.

void **canvas_set_shadow_texture_size**(size: int) 🔗

Sets the ProjectSettings.rendering/2d/shadow_atlas/size to use for Light2D shadow rendering (in pixels). The value is rounded up to the nearest power of 2.

RID **canvas_texture_create**() 🔗

Creates a canvas texture and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `canvas_texture_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method. See also texture_2d_create.

Note: The equivalent resource is CanvasTexture and is only meant to be used in 2D rendering, not 3D.

void **canvas_texture_set_channel**(canvas_texture: RID, channel: CanvasTextureChannel, texture: RID) 🔗

Sets the `channel`'s `texture` for the canvas texture specified by the `canvas_texture` RID. Equivalent to CanvasTexture.diffuse_texture, CanvasTexture.normal_texture and CanvasTexture.specular_texture.

void **canvas_texture_set_shading_parameters**(canvas_texture: RID, base_color: Color, shininess: float) 🔗

Sets the `base_color` and `shininess` to use for the canvas texture specified by the `canvas_texture` RID. Equivalent to CanvasTexture.specular_color and CanvasTexture.specular_shininess.

void **canvas_texture_set_texture_filter**(canvas_texture: RID, filter: CanvasItemTextureFilter)

Sets the texture `filter` mode to use for the canvas texture specified by the `canvas_texture` RID.

---

void **canvas_texture_set_texture_repeat**(canvas_texture: RID, repeat: CanvasItemTextureRepeat)

Sets the texture `repeat` mode to use for the canvas texture specified by the `canvas_texture` RID.

RID **compositor_create**()

Creates a new compositor and adds it to the RenderingServer. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

RID **compositor_effect_create**()

Creates a new rendering effect and adds it to the RenderingServer. It can be accessed with the RID that is returned.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

---

void **compositor_effect_set_callback**(effect: RID, callback_type: CompositorEffectCallbackType, callback: Callable)

Sets the callback type (`callback_type`) and callback method(`callback`) for this rendering effect.

---

void **compositor_effect_set_enabled**(effect: RID, enabled: bool)

Enables/disables this rendering effect.

---

void **compositor_effect_set_flag**(effect: RID, flag: CompositorEffectFlags, set: bool)

Sets the flag (`flag`) for this rendering effect to `true` or `false` (`set`).

---

void **compositor_set_compositor_effects**(compositor: RID, effects: Array[RID])

Sets the compositor effects for the specified compositor RID. `effects` should be an array containing RIDs created with compositor_effect_create.

RenderingDevice **create_local_rendering_device**() const

Creates a RenderingDevice that can be used to do draw and compute operations on a separate thread. Cannot draw to the screen nor share data with the global RenderingDevice.

Note: When using the OpenGL backend or when running in headless mode, this function always returns `null`.

Rect2 **debug_canvas_item_get_rect**(item: RID)

Returns the bounding rectangle for a canvas item in local space, as calculated by the renderer. This bound is used internally for culling.

Warning: This function is intended for debugging in the editor, and will pass through and return a zero Rect2 in exported projects.

RID **decal_create**()

Creates a decal and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `decal_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

To place in a scene, attach this decal to an instance using instance_set_base using the returned RID.

Note: The equivalent node is Decal.

---

void **decal_set_albedo_mix**(decal: RID, albedo_mix: float)

Sets the `albedo_mix` in the decal specified by the `decal` RID. Equivalent to Decal.albedo_mix.

---

void **decal_set_cull_mask**(decal: RID, mask: int)

Sets the cull `mask` in the decal specified by the `decal` RID. Equivalent to Decal.cull_mask.

---

void **decal_set_distance_fade**(decal: RID, enabled: bool, begin: float, length: float)

Sets the distance fade parameters in the decal specified by the `decal` RID. Equivalent to Decal.distance_fade_enabled, Decal.distance_fade_begin and Decal.distance_fade_length.

---

void **decal_set_emission_energy**(decal: RID, energy: float)

Sets the emission `energy` in the decal specified by the `decal` RID. Equivalent to Decal.emission_energy.

---

void **decal_set_fade**(decal: RID, above: float, below: float)

Sets the upper fade (`above`) and lower fade (`below`) in the decal specified by the `decal` RID. Equivalent to Decal.upper_fade and Decal.lower_fade.

void **decal_set_modulate**(decal: RID, color: Color)

Sets the color multiplier in the decal specified by the `decal` RID to `color`. Equivalent to Decal.modulate.

void **decal_set_normal_fade**(decal: RID, fade: float)

Sets the normal `fade` in the decal specified by the `decal` RID. Equivalent to Decal.normal_fade.

void **decal_set_size**(decal: RID, size: Vector3)

Sets the `size` of the decal specified by the `decal` RID. Equivalent to Decal.size.

void **decal_set_texture**(decal: RID, type: DecalTexture, texture: RID)

Sets the `texture` in the given texture `type` slot for the specified decal. Equivalent to Decal.set_texture.

void **decals_set_filter**(filter: DecalFilter)

Sets the texture `filter` mode to use when rendering decals. This parameter is global and cannot be set on a per-decal basis.

RID **directional_light_create**()

Creates a directional light and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID can be used in most `light_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

To place in a scene, attach this directional light to an instance using instance_set_base using the returned RID.

Note: The equivalent node is DirectionalLight3D.

void **directional_shadow_atlas_set_size**(size: int, is_16bits: bool)

Sets the `size` of the directional light shadows in 3D. See also ProjectSettings.rendering/lights_and_shadows/directional_shadow/size. This parameter is global and cannot be set on a per-viewport basis.

void **directional_soft_shadow_filter_set_quality**(quality: ShadowQuality)

Sets the filter `quality` for directional light shadows in 3D. See also ProjectSettings.rendering/lights_and_shadows/directional_shadow/soft_shadow_filter_quality. This parameter is global and cannot be set on a per-viewport basis.

Image **environment_bake_panorama**(environment: RID, bake_irradiance: bool, size: Vector2i)

Generates and returns an Image containing the radiance map for the specified `environment` RID's sky. This supports built-in sky material and custom sky shaders. If `bake_irradiance` is `true`, the irradiance map is saved instead of the radiance map. The radiance map is used to render reflected light, while the irradiance map is used to render ambient light. See also sky_bake_panorama.

Note: The image is saved in linear color space without any tonemapping performed, which means it will look too dark if viewed directly in an image editor.

Note: `size` should be a 2:1 aspect ratio for the generated panorama to have square pixels. For radiance maps, there is no point in using a height greater than Sky.radiance_size, as it won't increase detail. Irradiance maps only contain low-frequency data, so there is usually no point in going past a size of 128×64 pixels when saving an irradiance map.

RID **environment_create**()

Creates an environment and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `environment_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is Environment.

void **environment_glow_set_use_bicubic_upscale**(enable: bool)

If `enable` is `true`, enables bicubic upscaling for glow which improves quality at the cost of performance. Equivalent to ProjectSettings.rendering/environment/glow/upscale_mode.

void **environment_set_adjustment**(env: RID, enable: bool, brightness: float, contrast: float, saturation: float, use_1d_color_correction: bool, color_correction: RID)

Sets the values to be used with the "adjustments" post-process effect. See Environment for more details.

void **environment_set_ambient_light**(env: RID, color: Color, ambient: EnvironmentAmbientSource = 0, energy: float = 1.0, sky_contibution: float = 0.0, reflection_source: EnvironmentReflectionSource = 0)

Sets the values to be used for ambient light rendering. See Environment for more details.

void **environment_set_background**(env: RID, bg: EnvironmentBG) 🔗

Sets the environment's background mode. Equivalent to Environment.background_mode.

---

void **environment_set_bg_color**(env: RID, color: Color) 🔗

Color displayed for clear areas of the scene. Only effective if using the ENV_BG_COLOR background mode.

---

void **environment_set_bg_energy**(env: RID, multiplier: float, exposure_value: float) 🔗

Sets the intensity of the background color.

---

void **environment_set_canvas_max_layer**(env: RID, max_layer: int) 🔗

Sets the maximum layer to use if using Canvas background mode.

---

void **environment_set_fog**(env: RID, enable: bool, light_color: Color, light_energy: float, sun_scatter: float, density: float, height: float, height_density: float, aerial_perspective: float, sky_affect: float, fog_mode: EnvironmentFogMode = 0) 🔗

Configures fog for the specified environment RID. See `fog_*` properties in Environment for more information.

---

void **environment_set_glow**(env: RID, enable: bool, levels: PackedFloat32Array, intensity: float, strength: float, mix: float, bloom_threshold: float, blend_mode: EnvironmentGlowBlendMode, hdr_bleed_threshold: float, hdr_bleed_scale: float, hdr_luminance_cap: float, glow_map_strength: float, glow_map: RID) 🔗

Configures glow for the specified environment RID. See `glow_*` properties in Environment for more information.

---

void **environment_set_sdfgi**(env: RID, enable: bool, cascades: int, min_cell_size: float, y_scale: EnvironmentSDFGIYScale, use_occlusion: bool, bounce_feedback: float, read_sky: bool, energy: float, normal_bias: float, probe_bias: float) 🔗

Configures signed distance field global illumination for the specified environment RID. See `sdfgi_*` properties in Environment for more information.

---

void **environment_set_sdfgi_frames_to_converge**(frames: EnvironmentSDFGIFramesToConverge) 🔗

Sets the number of frames to use for converging signed distance field global illumination. Equivalent to ProjectSettings.rendering/global_illumination/sdfgi/frames_to_converge.

---

void **environment_set_sdfgi_frames_to_update_light**(frames: EnvironmentSDFGIFramesToUpdateLight)

Sets the update speed for dynamic lights' indirect lighting when computing signed distance field global illumination. Equivalent to ProjectSettings.rendering/global_illumination/sdfgi/frames_to_update_lights.

---

void **environment_set_sdfgi_ray_count**(ray_count: EnvironmentSDFGIRayCount)

Sets the number of rays to throw per frame when computing signed distance field global illumination. Equivalent to ProjectSettings.rendering/global_illumination/sdfgi/probe_ray_count.

---

void **environment_set_sky**(env: RID, sky: RID)

Sets the Sky to be used as the environment's background when using *BGMode* sky. Equivalent to Environment.sky.

---

void **environment_set_sky_custom_fov**(env: RID, scale: float)

Sets a custom field of view for the background Sky. Equivalent to Environment.sky_custom_fov.

---

void **environment_set_sky_orientation**(env: RID, orientation: Basis)

Sets the rotation of the background Sky expressed as a Basis. Equivalent to Environment.sky_rotation, where the rotation vector is used to construct the Basis.

---

void **environment_set_ssao**(env: RID, enable: bool, radius: float, intensity: float, power: float, detail: float, horizon: float, sharpness: float, light_affect: float, ao_channel_affect: float)

Sets the variables to be used with the screen-space ambient occlusion (SSAO) post-process effect. See Environment for more details.

---

void **environment_set_ssao_quality**(quality: EnvironmentSSAOQuality, half_size: bool, adaptive_target: float, blur_passes: int, fadeout_from: float, fadeout_to: float)

Sets the quality level of the screen-space ambient occlusion (SSAO) post-process effect. See Environment for more details.

---

void **environment_set_ssil_quality**(quality: EnvironmentSSILQuality, half_size: bool, adaptive_target: float, blur_passes: int, fadeout_from: float, fadeout_to: float)

Sets the quality level of the screen-space indirect lighting (SSIL) post-process effect. See Environment for more details.

---

void **environment_set_ssr**(env: RID, enable: bool, max_steps: int, fade_in: float, fade_out: float, depth_tolerance: float)

Sets the variables to be used with the screen-space reflections (SSR) post-process effect. See Environment for more details.

---

void **environment_set_ssr_roughness_quality**(quality: EnvironmentSSRRoughnessQuality)

There is currently no description for this method. Please help us by contributing one!

---

void **environment_set_tonemap**(env: RID, tone_mapper: EnvironmentToneMapper, exposure: float, white: float)

Sets the variables to be used with the "tonemap" post-process effect. See Environment for more details.

---

void **environment_set_volumetric_fog**(env: RID, enable: bool, density: float, albedo: Color, emission: Color, emission_energy: float, anisotropy: float, length: float, p_detail_spread: float, gi_inject: float, temporal_reprojection: bool, temporal_reprojection_amount: float, ambient_inject: float, sky_affect: float)

Sets the variables to be used with the volumetric fog post-process effect. See Environment for more details.

---

void **environment_set_volumetric_fog_filter_active**(active: bool)

Enables filtering of the volumetric fog scattering buffer. This results in much smoother volumes with very few under-sampling artifacts.

---

void **environment_set_volumetric_fog_volume_size**(size: int, depth: int)

Sets the resolution of the volumetric fog's froxel buffer. `size` is modified by the screen's aspect ratio and then used to set the width and height of the buffer. While `depth` is directly used to set the depth of the buffer.

RID **fog_volume_create**()

Creates a new fog volume and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `fog_volume_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent node is FogVolume.

---

void **fog_volume_set_material**(fog_volume: RID, material: RID)

Sets the Material of the fog volume. Can be either a FogMaterial or a custom ShaderMaterial.

void **fog_volume_set_shape**(fog_volume: RID, shape: FogVolumeShape)

Sets the shape of the fog volume to either FOG_VOLUME_SHAPE_ELLIPSOID, FOG_VOLUME_SHAPE_CONE, FOG_VOLUME_SHAPE_CYLINDER, FOG_VOLUME_SHAPE_BOX or FOG_VOLUME_SHAPE_WORLD.

void **fog_volume_set_size**(fog_volume: RID, size: Vector3)

Sets the size of the fog volume when shape is FOG_VOLUME_SHAPE_ELLIPSOID, FOG_VOLUME_SHAPE_CONE, FOG_VOLUME_SHAPE_CYLINDER or FOG_VOLUME_SHAPE_BOX.

void **force_draw**(swap_buffers: bool = true, frame_step: float = 0.0)

Forces redrawing of all viewports at once. Must be called from the main thread.

void **force_sync**()

Forces a synchronization between the CPU and GPU, which may be required in certain cases. Only call this when needed, as CPU-GPU synchronization has a performance cost.

void **free_rid**(rid: RID)

Tries to free an object in the RenderingServer. To avoid memory leaks, this should be called after using an object as memory management does not occur automatically when using RenderingServer directly.

Color **get_default_clear_color**()

Returns the default clear color which is used when a specific clear color has not been selected. See also set_default_clear_color.

float **get_frame_setup_time_cpu**() const

Returns the time taken to setup rendering on the CPU in milliseconds. This value is shared across all viewports and does *not* require viewport_set_measure_render_time to be enabled on a viewport to be queried. See also viewport_get_measured_render_time_cpu.

RenderingDevice **get_rendering_device**() const

Returns the global RenderingDevice.

Note: When using the OpenGL backend or when running in headless mode, this function always returns `null`.

int **get_rendering_info**(info: RenderingInfo) 🔗

Returns a statistic about the rendering engine which can be used for performance profiling. See RenderingInfo for a list of values that can be queried. See also viewport_get_render_info, which returns information specific to a viewport.

Note: Only 3D rendering is currently taken into account by some of these values, such as the number of draw calls.

Note: Rendering information is not available until at least 2 frames have been rendered by the engine. If rendering information is not available, get_rendering_info returns `0`. To print rendering information in `_ready()` successfully, use the following:

```
func _ready():
    for _i in 2:
        await get_tree().process_frame

    print(RenderingServer.get_rendering_info(RENDERING_INFO_TOTAL_DRAW_CALLS_IN_FRAME))
```

Array[Dictionary] **get_shader_parameter_list**(shader: RID) const 🔗

Returns the parameters of a shader.

RID **get_test_cube**() 🔗

Returns the RID of the test cube. This mesh will be created and returned on the first call to get_test_cube, then it will be cached for subsequent calls. See also make_sphere_mesh.

RID **get_test_texture**() 🔗

Returns the RID of a 256×256 texture with a testing pattern on it (in Image.FORMAT_RGB8 format). This texture will be created and returned on the first call to get_test_texture, then it will be cached for subsequent calls. See also get_white_texture.

Example of getting the test texture and applying it to a Sprite2D node:

```
var texture_rid = RenderingServer.get_test_texture()
var texture = ImageTexture.create_from_image(RenderingServer.texture_2d_get(texture_rid))
$Sprite2D.texture = texture
```

String **get_video_adapter_api_version**() const 🔗

Returns the version of the graphics video adapter *currently in use* (e.g. "1.2.189" for Vulkan, "3.3.0 NVIDIA 510.60.02" for OpenGL). This version may be different from the actual latest version supported by the hardware, as Godot may not always request the latest version. See also OS.get_video_adapter_driver_info.

Note: When running a headless or server binary, this function returns an empty string.

String **get_video_adapter_name**() const 🔗

Returns the name of the video adapter (e.g. "GeForce GTX 1080/PCIe/SSE2").

Note: When running a headless or server binary, this function returns an empty string.

Note: On the web platform, some browsers such as Firefox may report a different, fixed GPU name such as "GeForce GTX 980" (regardless of the user's actual GPU model). This is done to make fingerprinting more difficult.

DeviceType **get_video_adapter_type**() const

Returns the type of the video adapter. Since dedicated graphics cards from a given generation will *usually* be significantly faster than integrated graphics made in the same generation, the device type can be used as a basis for automatic graphics settings adjustment. However, this is not always true, so make sure to provide users with a way to manually override graphics settings.

Note: When using the OpenGL backend or when running in headless mode, this function always returns RenderingDevice.DEVICE_TYPE_OTHER.

String **get_video_adapter_vendor**() const

Returns the vendor of the video adapter (e.g. "NVIDIA Corporation").

Note: When running a headless or server binary, this function returns an empty string.

RID **get_white_texture**()

Returns the ID of a 4×4 white texture (in Image.FORMAT_RGB8 format). This texture will be created and returned on the first call to get_white_texture, then it will be cached for subsequent calls. See also get_test_texture.

Example of getting the white texture and applying it to a Sprite2D node:

```
var texture_rid = RenderingServer.get_white_texture()
var texture = ImageTexture.create_from_image(RenderingServer.texture_2d_get(texture_rid))
$Sprite2D.texture = texture
```

---

void **gi_set_use_half_resolution**(half_resolution: bool)

If `half_resolution` is `true`, renders VoxelGI and SDFGI (Environment.sdfgi_enabled) buffers at halved resolution on each axis (e.g. 960×540 when the viewport size is 1920×1080). This improves performance significantly when VoxelGI or SDFGI is enabled, at the cost of artifacts that may be visible on polygon edges. The loss in quality becomes less noticeable as the viewport resolution increases. LightmapGI rendering is not affected by this setting. Equivalent to ProjectSettings.rendering/global_illumination/gi/use_half_resolution.

---

void **global_shader_parameter_add**(name: StringName, type: GlobalShaderParameterType, default_value: Variant)

Creates a new global shader uniform.

Note: Global shader parameter names are case-sensitive.

Variant **global_shader_parameter_get**(name: StringName) const 🔗

Returns the value of the global shader uniform specified by `name`.

Note: global_shader_parameter_get has a large performance penalty as the rendering thread needs to synchronize with the calling thread, which is slow. Do not use this method during gameplay to avoid stuttering. If you need to read values in a script after setting them, consider creating an autoload where you store the values you need to query at the same time you're setting them as global parameters.

Array[StringName] **global_shader_parameter_get_list**() const 🔗

Returns the list of global shader uniform names.

Note: global_shader_parameter_get has a large performance penalty as the rendering thread needs to synchronize with the calling thread, which is slow. Do not use this method during gameplay to avoid stuttering. If you need to read values in a script after setting them, consider creating an autoload where you store the values you need to query at the same time you're setting them as global parameters.

GlobalShaderParameterType **global_shader_parameter_get_type**(name: StringName) const 🔗

Returns the type associated to the global shader uniform specified by `name`.

Note: global_shader_parameter_get has a large performance penalty as the rendering thread needs to synchronize with the calling thread, which is slow. Do not use this method during gameplay to avoid stuttering. If you need to read values in a script after setting them, consider creating an autoload where you store the values you need to query at the same time you're setting them as global parameters.

---

void **global_shader_parameter_remove**(name: StringName) 🔗

Removes the global shader uniform specified by `name`.

---

void **global_shader_parameter_set**(name: StringName, value: Variant) 🔗

Sets the global shader uniform `name` to `value`.

---

void **global_shader_parameter_set_override**(name: StringName, value: Variant) 🔗

Overrides the global shader uniform `name` with `value`. Equivalent to the ShaderGlobalsOverride node.

---

bool **has_changed**() const 🔗

Returns `true` if changes have been made to the RenderingServer's data. force_draw is usually called if this happens.

---

bool **has_feature**(feature: Features) const 🔗

Deprecated: This method has not been used since Godot 3.0.

This method does nothing and always returns `false`.

---

bool **has_os_feature**(feature: String) const 🔗

Returns `true` if the OS supports a certain `feature`. Features might be `s3tc`, `etc`, and `etc2`.

---

void **instance_attach_object_instance_id**(instance: RID, id: int) 🔗

Attaches a unique Object ID to instance. Object ID must be attached to instance for proper culling with instances_cull_aabb, instances_cull_convex, and instances_cull_ray.

---

void **instance_attach_skeleton**(instance: RID, skeleton: RID) 🔗

Attaches a skeleton to an instance. Removes the previous skeleton from the instance.

RID **instance_create**() 🔗

Creates a visual instance and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `instance_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

An instance is a way of placing a 3D object in the scenario. Objects like particles, meshes, reflection probes and decals need to be associated with an instance to be visible in the scenario using instance_set_base.

Note: The equivalent node is VisualInstance3D.

RID **instance_create2**(base: RID, scenario: RID) 🔗

Creates a visual instance, adds it to the RenderingServer, and sets both base and scenario. It can be accessed with the RID that is returned. This RID will be used in all `instance_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method. This is a shorthand for using instance_create and setting the base and scenario manually.

Variant **instance_geometry_get_shader_parameter**(instance: RID, parameter: StringName) const 🔗

Returns the value of the per-instance shader uniform from the specified 3D geometry instance. Equivalent to GeometryInstance3D.get_instance_shader_parameter.

Note: Per-instance shader parameter names are case-sensitive.

Variant **instance_geometry_get_shader_parameter_default_value**(instance: RID, parameter: StringName) const

Returns the default value of the per-instance shader uniform from the specified 3D geometry instance. Equivalent to GeometryInstance3D.get_instance_shader_parameter.

Array[Dictionary] **instance_geometry_get_shader_parameter_list**(instance: RID) const

Returns a dictionary of per-instance shader uniform names of the per-instance shader uniform from the specified 3D geometry instance. The returned dictionary is in PropertyInfo format, with the keys `name`, `class_name`, `type`, `hint`, `hint_string` and `usage`. Equivalent to GeometryInstance3D.get_instance_shader_parameter.

---

void **instance_geometry_set_cast_shadows_setting**(instance: RID, shadow_casting_setting: ShadowCastingSetting)

Sets the shadow casting setting to one of ShadowCastingSetting. Equivalent to GeometryInstance3D.cast_shadow.

---

void **instance_geometry_set_flag**(instance: RID, flag: InstanceFlags, enabled: bool)

Sets the flag for a given InstanceFlags. See InstanceFlags for more details.

---

void **instance_geometry_set_lightmap**(instance: RID, lightmap: RID, lightmap_uv_scale: Rect2, lightmap_slice: int)

Sets the lightmap GI instance to use for the specified 3D geometry instance. The lightmap UV scale for the specified instance (equivalent to GeometryInstance3D.gi_lightmap_scale) and lightmap atlas slice must also be specified.

---

void **instance_geometry_set_lod_bias**(instance: RID, lod_bias: float)

Sets the level of detail bias to use when rendering the specified 3D geometry instance. Higher values result in higher detail from further away. Equivalent to GeometryInstance3D.lod_bias.

---

void **instance_geometry_set_material_overlay**(instance: RID, material: RID)

Sets a material that will be rendered for all surfaces on top of active materials for the mesh associated with this instance. Equivalent to GeometryInstance3D.material_overlay.

---

void **instance_geometry_set_material_override**(instance: RID, material: RID)

Sets a material that will override the material for all surfaces on the mesh associated with this instance. Equivalent to GeometryInstance3D.material_override.

---

void **instance_geometry_set_shader_parameter**(instance: RID, parameter: StringName, value: Variant)

Sets the per-instance shader uniform on the specified 3D geometry instance. Equivalent to GeometryInstance3D.set_instance_shader_parameter.

---

void **instance_geometry_set_transparency**(instance: RID, transparency: float)

Sets the transparency for the given geometry instance. Equivalent to GeometryInstance3D.transparency.

A transparency of `0.0` is fully opaque, while `1.0` is fully transparent. Values greater than `0.0` (exclusive) will force the geometry's materials to go through the transparent pipeline, which is slower to render and can exhibit rendering issues due to incorrect transparency sorting. However, unlike using a transparent material, setting `transparency` to a value greater than `0.0` (exclusive) will *not* disable shadow rendering.

In spatial shaders, `1.0 - transparency` is set as the default value of the `ALPHA` built-in.

Note: `transparency` is clamped between `0.0` and `1.0`, so this property cannot be used to make transparent materials more opaque than they originally are.

---

void **instance_geometry_set_visibility_range**(instance: RID, min: float, max: float, min_margin: float, max_margin: float, fade_mode: VisibilityRangeFadeMode)

Sets the visibility range values for the given geometry instance. Equivalent to GeometryInstance3D.visibility_range_begin and related properties.

---

void **instance_set_base**(instance: RID, base: RID)

Sets the base of the instance. A base can be any of the 3D objects that are created in the RenderingServer that can be displayed. For example, any of the light types, mesh, multimesh, particle system, reflection probe, decal, lightmap, voxel GI and visibility notifiers are all types that can be set as the base of an instance in order to be displayed in the scenario.

---

void **instance_set_blend_shape_weight**(instance: RID, shape: int, weight: float)

Sets the weight for a given blend shape associated with this instance.

---

void **instance_set_custom_aabb**(instance: RID, aabb: AABB)

Sets a custom AABB to use when culling objects from the view frustum. Equivalent to setting GeometryInstance3D.custom_aabb.

void **instance_set_extra_visibility_margin**(instance: RID, margin: float)

Sets a margin to increase the size of the AABB when culling objects from the view frustum. This allows you to avoid culling objects that fall outside the view frustum. Equivalent to GeometryInstance3D.extra_cull_margin.

void **instance_set_ignore_culling**(instance: RID, enabled: bool)

If `true`, ignores both frustum and occlusion culling on the specified 3D geometry instance. This is not the same as GeometryInstance3D.ignore_occlusion_culling, which only ignores occlusion culling and leaves frustum culling intact.

void **instance_set_layer_mask**(instance: RID, mask: int)

Sets the render layers that this instance will be drawn to. Equivalent to VisualInstance3D.layers.

void **instance_set_pivot_data**(instance: RID, sorting_offset: float, use_aabb_center: bool)

Sets the sorting offset and switches between using the bounding box or instance origin for depth sorting.

void **instance_set_scenario**(instance: RID, scenario: RID)

Sets the scenario that the instance is in. The scenario is the 3D world that the objects will be displayed in.

void **instance_set_surface_override_material**(instance: RID, surface: int, material: RID)

Sets the override material of a specific surface. Equivalent to MeshInstance3D.set_surface_override_material.

void **instance_set_transform**(instance: RID, transform: Transform3D)

Sets the world space transform of the instance. Equivalent to Node3D.global_transform.

void **instance_set_visibility_parent**(instance: RID, parent: RID)

Sets the visibility parent for the given instance. Equivalent to Node3D.visibility_parent.

void **instance_set_visible**(instance: RID, visible: bool)

Sets whether an instance is drawn or not. Equivalent to Node3D.visible.

PackedInt64Array **instances_cull_aabb**(aabb: AABB, scenario: RID = RID()) const

Returns an array of object IDs intersecting with the provided AABB. Only 3D nodes that inherit from VisualInstance3D are considered, such as MeshInstance3D or DirectionalLight3D. Use @GlobalScope.instance_from_id to obtain the actual nodes. A scenario RID must be provided, which is available in the World3D you want to query. This forces an update for all resources queued to update.

Warning: This function is primarily intended for editor usage. For in-game use cases, prefer physics collision.

PackedInt64Array **instances_cull_convex**(convex: Array[Plane], scenario: RID = RID()) const

Returns an array of object IDs intersecting with the provided convex shape. Only 3D nodes that inherit from VisualInstance3D are considered, such as MeshInstance3D or DirectionalLight3D. Use @GlobalScope.instance_from_id to obtain the actual nodes. A scenario RID must be provided, which is available in the World3D you want to query. This forces an update for all resources queued to update.

Warning: This function is primarily intended for editor usage. For in-game use cases, prefer physics collision.

PackedInt64Array **instances_cull_ray**(from: Vector3, to: Vector3, scenario: RID = RID()) const

Returns an array of object IDs intersecting with the provided 3D ray. Only 3D nodes that inherit from VisualInstance3D are considered, such as MeshInstance3D or DirectionalLight3D. Use @GlobalScope.instance_from_id to obtain the actual nodes. A scenario RID must be provided, which is available in the World3D you want to query. This forces an update for all resources queued to update.

Warning: This function is primarily intended for editor usage. For in-game use cases, prefer physics collision.

---

bool **is_on_render_thread**()

Returns `true` if our code is currently executing on the rendering thread.

---

void **light_directional_set_blend_splits**(light: RID, enable: bool)

If `true`, this directional light will blend between shadow map splits resulting in a smoother transition between them. Equivalent to DirectionalLight3D.directional_shadow_blend_splits.

---

void **light_directional_set_shadow_mode**(light: RID, mode: LightDirectionalShadowMode)

Sets the shadow mode for this directional light. Equivalent to DirectionalLight3D.directional_shadow_mode. See LightDirectionalShadowMode for options.

---

void **light_directional_set_sky_mode**(light: RID, mode: LightDirectionalSkyMode)

If `true`, this light will not be used for anything except sky shaders. Use this for lights that impact your sky shader that you may want to hide from affecting the rest of the scene. For example, you may want to enable this when the sun in your sky shader falls below the horizon.

---

void **light_omni_set_shadow_mode**(light: RID, mode: LightOmniShadowMode)

Sets whether to use a dual paraboloid or a cubemap for the shadow map. Dual paraboloid is faster but may suffer from artifacts. Equivalent to OmniLight3D.omni_shadow_mode.

---

void **light_projectors_set_filter**(filter: LightProjectorFilter)

Sets the texture filter mode to use when rendering light projectors. This parameter is global and cannot be set on a per-light basis.

---

void **light_set_bake_mode**(light: RID, bake_mode: LightBakeMode)

Sets the bake mode to use for the specified 3D light. Equivalent to Light3D.light_bake_mode.

---

void **light_set_color**(light: RID, color: Color)

Sets the color of the light. Equivalent to Light3D.light_color.

---

void **light_set_cull_mask**(light: RID, mask: int)

Sets the cull mask for this 3D light. Lights only affect objects in the selected layers. Equivalent to Light3D.light_cull_mask.

---

void **light_set_distance_fade**(decal: RID, enabled: bool, begin: float, shadow: float, length: float)

Sets the distance fade for this 3D light. This acts as a form of level of detail (LOD) and can be used to improve performance. Equivalent to Light3D.distance_fade_enabled, Light3D.distance_fade_begin, Light3D.distance_fade_shadow, and Light3D.distance_fade_length.

---

void **light_set_max_sdfgi_cascade**(light: RID, cascade: int)

Sets the maximum SDFGI cascade in which the 3D light's indirect lighting is rendered. Higher values allow the light to be rendered in SDFGI further away from the camera.

---

void **light_set_negative**(light: RID, enable: bool)

If `true`, the 3D light will subtract light instead of adding light. Equivalent to Light3D.light_negative.

---

void **light_set_param**(light: RID, param: LightParam, value: float)

Sets the specified 3D light parameter. See LightParam for options. Equivalent to Light3D.set_param.

void **light_set_projector**(light: RID, texture: RID)

Sets the projector texture to use for the specified 3D light. Equivalent to Light3D.light_projector.

void **light_set_reverse_cull_face_mode**(light: RID, enabled: bool)

If `true`, reverses the backface culling of the mesh. This can be useful when you have a flat mesh that has a light behind it. If you need to cast a shadow on both sides of the mesh, set the mesh to use double-sided shadows with instance_geometry_set_cast_shadows_setting. Equivalent to Light3D.shadow_reverse_cull_face.

void **light_set_shadow**(light: RID, enabled: bool)

If `true`, light will cast shadows. Equivalent to Light3D.shadow_enabled.

RID **lightmap_create**()

Creates a new lightmap global illumination instance and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `lightmap_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent node is LightmapGI.

PackedInt32Array **lightmap_get_probe_capture_bsp_tree**(lightmap: RID) const

There is currently no description for this method. Please help us by contributing one!

PackedVector3Array **lightmap_get_probe_capture_points**(lightmap: RID) const

There is currently no description for this method. Please help us by contributing one!

PackedColorArray **lightmap_get_probe_capture_sh**(lightmap: RID) const

There is currently no description for this method. Please help us by contributing one!

PackedInt32Array **lightmap_get_probe_capture_tetrahedra**(lightmap: RID) const

There is currently no description for this method. Please help us by contributing one!

void **lightmap_set_baked_exposure_normalization**(lightmap: RID, baked_exposure: float)

Used to inform the renderer what exposure normalization value was used while baking the lightmap. This value will be used and modulated at run time to ensure that the lightmap maintains a consistent level of exposure even if the scene-wide exposure normalization is changed at run time. For more information see camera_attributes_set_exposure.

---

void **lightmap_set_probe_bounds**(lightmap: RID, bounds: AABB)

There is currently no description for this method. Please help us by contributing one!

---

void **lightmap_set_probe_capture_data**(lightmap: RID, points: PackedVector3Array, point_sh: PackedColorArray, tetrahedra: PackedInt32Array, bsp_tree: PackedInt32Array)

There is currently no description for this method. Please help us by contributing one!

---

void **lightmap_set_probe_capture_update_speed**(speed: float)

There is currently no description for this method. Please help us by contributing one!

---

void **lightmap_set_probe_interior**(lightmap: RID, interior: bool)

There is currently no description for this method. Please help us by contributing one!

---

void **lightmap_set_textures**(lightmap: RID, light: RID, uses_sh: bool)

Set the textures on the given `lightmap` GI instance to the texture array pointed to by the `light` RID. If the lightmap texture was baked with LightmapGI.directional set to `true`, then `uses_sh` must also be `true`.

RID **make_sphere_mesh**(latitudes: int, longitudes: int, radius: float)

Returns a mesh of a sphere with the given number of horizontal subdivisions, vertical subdivisions and radius. See also get_test_cube.

RID **material_create**()

Creates an empty material and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `material_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is Material.

Variant **material_get_param**(material: RID, parameter: StringName) const

Returns the value of a certain material's parameter.

---

void **material_set_next_pass**(material: RID, next_material: RID)

Sets an object's next material.

---

void **material_set_param**(material: RID, parameter: StringName, value: Variant)

Sets a material's parameter.

---

void **material_set_render_priority**(material: RID, priority: int)

Sets a material's render priority.

---

void **material_set_shader**(shader_material: RID, shader: RID)

Sets a shader material's shader.

---

void **mesh_add_surface**(mesh: RID, surface: Dictionary)

There is currently no description for this method. Please help us by contributing one!

---

void **mesh_add_surface_from_arrays**(mesh: RID, primitive: PrimitiveType, arrays: Array, blend_shapes: Array = [], lods: Dictionary = {}, compress_format: BitField[ArrayFormat] = 0)

There is currently no description for this method. Please help us by contributing one!

---

void **mesh_clear**(mesh: RID)

Removes all surfaces from a mesh.

RID **mesh_create**()

Creates a new mesh and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `mesh_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

To place in a scene, attach this mesh to an instance using instance_set_base using the returned RID.

Note: The equivalent resource is Mesh.

RID **mesh_create_from_surfaces**(surfaces: Array[Dictionary], blend_shape_count: int = 0)

There is currently no description for this method. Please help us by contributing one!

---

int **mesh_get_blend_shape_count**(mesh: RID) const

Returns a mesh's blend shape count.

BlendShapeMode **mesh_get_blend_shape_mode**(mesh: RID) const

Returns a mesh's blend shape mode.

AABB **mesh_get_custom_aabb**(mesh: RID) const

Returns a mesh's custom aabb.

Dictionary **mesh_get_surface**(mesh: RID, surface: int)

There is currently no description for this method. Please help us by contributing one!

---

int **mesh_get_surface_count**(mesh: RID) const

Returns a mesh's number of surfaces.

---

void **mesh_set_blend_shape_mode**(mesh: RID, mode: BlendShapeMode)

Sets a mesh's blend shape mode.

---

void **mesh_set_custom_aabb**(mesh: RID, aabb: AABB)

Sets a mesh's custom aabb.

---

void **mesh_set_shadow_mesh**(mesh: RID, shadow_mesh: RID)

There is currently no description for this method. Please help us by contributing one!

Array **mesh_surface_get_arrays**(mesh: RID, surface: int) const

Returns a mesh's surface's buffer arrays.

Array[Array] **mesh_surface_get_blend_shape_arrays**(mesh: RID, surface: int) const

Returns a mesh's surface's arrays for blend shapes.

---

int **mesh_surface_get_format_attribute_stride**(format: BitField[ArrayFormat], vertex_count: int) const

Returns the stride of the attribute buffer for a mesh with given `format`.

---

int **mesh_surface_get_format_normal_tangent_stride**(format: BitField[ArrayFormat], vertex_count: int) const

Returns the stride of the combined normals and tangents for a mesh with given `format`. Note importantly that, while normals and tangents are in the vertex buffer with vertices, they are only interleaved with each other and so have a different stride than vertex positions.

---

int **mesh_surface_get_format_offset**(format: BitField[ArrayFormat], vertex_count: int, array_index: int) const

Returns the offset of a given attribute by `array_index` in the start of its respective buffer.

---

int **mesh_surface_get_format_skin_stride**(format: BitField[ArrayFormat], vertex_count: int) const

Returns the stride of the skin buffer for a mesh with given `format`.

---

int **mesh_surface_get_format_vertex_stride**(format: BitField[ArrayFormat], vertex_count: int) const

Returns the stride of the vertex positions for a mesh with given `format`. Note importantly that vertex positions are stored consecutively and are not interleaved with the other attributes in the vertex buffer (normals and tangents).

RID **mesh_surface_get_material**(mesh: RID, surface: int) const

Returns a mesh's surface's material.

---

void **mesh_surface_set_material**(mesh: RID, surface: int, material: RID)

Sets a mesh's surface's material.

---

void **mesh_surface_update_attribute_region**(mesh: RID, surface: int, offset: int, data: PackedByteArray)

There is currently no description for this method. Please help us by contributing one!

---

void **mesh_surface_update_skin_region**(mesh: RID, surface: int, offset: int, data: PackedByteArray)

There is currently no description for this method. Please help us by contributing one!

---

void **mesh_surface_update_vertex_region**(mesh: RID, surface: int, offset: int, data: PackedByteArray)

There is currently no description for this method. Please help us by contributing one!

---

void **multimesh_allocate_data**(multimesh: RID, instances: int, transform_format: MultimeshTransformFormat, color_format: bool = false, custom_data_format: bool = false)

There is currently no description for this method. Please help us by contributing one!

RID **multimesh_create**()

Creates a new multimesh on the RenderingServer and returns an RID handle. This RID will be used in all `multimesh_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

To place in a scene, attach this multimesh to an instance using instance_set_base using the returned RID.

Note: The equivalent resource is MultiMesh.

AABB **multimesh_get_aabb**(multimesh: RID) const

Calculates and returns the axis-aligned bounding box that encloses all instances within the multimesh.

PackedFloat32Array **multimesh_get_buffer**(multimesh: RID) const

Returns the MultiMesh data (such as instance transforms, colors, etc.). See multimesh_set_buffer for details on the returned data.

Note: If the buffer is in the engine's internal cache, it will have to be fetched from GPU memory and possibly decompressed. This means multimesh_get_buffer is potentially a slow operation and should be avoided whenever possible.

AABB **multimesh_get_custom_aabb**(multimesh: RID) const

Returns the custom AABB defined for this MultiMesh resource.

---

int **multimesh_get_instance_count**(multimesh: RID) const

Returns the number of instances allocated for this multimesh.

RID **multimesh_get_mesh**(multimesh: RID) const

Returns the RID of the mesh that will be used in drawing this multimesh.

int **multimesh_get_visible_instances**(multimesh: RID) const

Returns the number of visible instances for this multimesh.

Color **multimesh_instance_get_color**(multimesh: RID, index: int) const

Returns the color by which the specified instance will be modulated.

Color **multimesh_instance_get_custom_data**(multimesh: RID, index: int) const

Returns the custom data associated with the specified instance.

Transform3D **multimesh_instance_get_transform**(multimesh: RID, index: int) const

Returns the Transform3D of the specified instance.

Transform2D **multimesh_instance_get_transform_2d**(multimesh: RID, index: int) const

Returns the Transform2D of the specified instance. For use when the multimesh is set to use 2D transforms.

void **multimesh_instance_set_color**(multimesh: RID, index: int, color: Color)

Sets the color by which this instance will be modulated. Equivalent to MultiMesh.set_instance_color.

void **multimesh_instance_set_custom_data**(multimesh: RID, index: int, custom_data: Color)

Sets the custom data for this instance. Custom data is passed as a Color, but is interpreted as a `vec4` in the shader. Equivalent to MultiMesh.set_instance_custom_data.

void **multimesh_instance_set_transform**(multimesh: RID, index: int, transform: Transform3D)

Sets the Transform3D for this instance. Equivalent to MultiMesh.set_instance_transform.

void **multimesh_instance_set_transform_2d**(multimesh: RID, index: int, transform: Transform2D)

Sets the Transform2D for this instance. For use when multimesh is used in 2D. Equivalent to MultiMesh.set_instance_transform_2d.

---

void **multimesh_set_buffer**(multimesh: RID, buffer: PackedFloat32Array) 🔗

Set the entire data to use for drawing the `multimesh` at once to `buffer` (such as instance transforms and colors). `buffer`'s size must match the number of instances multiplied by the per-instance data size (which depends on the enabled MultiMesh fields). Otherwise, an error message is printed and nothing is rendered. See also multimesh_get_buffer.

The per-instance data size and expected data order is:

```
2D:
  - Position: 8 floats (8 floats for Transform2D)
  - Position + Vertex color: 12 floats (8 floats for Transform2D, 4 floats for Color)
  - Position + Custom data: 12 floats (8 floats for Transform2D, 4 floats of custom data)
  - Position + Vertex color + Custom data: 16 floats (8 floats for Transform2D, 4 floats for
Color, 4 floats of custom data)
3D:
  - Position: 12 floats (12 floats for Transform3D)
  - Position + Vertex color: 16 floats (12 floats for Transform3D, 4 floats for Color)
  - Position + Custom data: 16 floats (12 floats for Transform3D, 4 floats of custom data)
  - Position + Vertex color + Custom data: 20 floats (12 floats for Transform3D, 4 floats for
Color, 4 floats of custom data)
```

---

void **multimesh_set_custom_aabb**(multimesh: RID, aabb: AABB) 🔗

Sets the custom AABB for this MultiMesh resource.

---

void **multimesh_set_mesh**(multimesh: RID, mesh: RID) 🔗

Sets the mesh to be drawn by the multimesh. Equivalent to MultiMesh.mesh.

---

void **multimesh_set_visible_instances**(multimesh: RID, visible: int) 🔗

Sets the number of instances visible at a given time. If -1, all instances that have been allocated are drawn. Equivalent to MultiMesh.visible_instance_count.

RID **occluder_create**() 🔗

Creates an occluder instance and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `occluder_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is Occluder3D (not to be confused with the OccluderInstance3D node).

---

void **occluder_set_mesh**(occluder: RID, vertices: PackedVector3Array, indices: PackedInt32Array) 🔗

Sets the mesh data for the given occluder RID, which controls the shape of the occlusion culling that will be performed.

RID **omni_light_create**()

Creates a new omni light and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID can be used in most `light_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

To place in a scene, attach this omni light to an instance using instance_set_base using the returned RID.

Note: The equivalent node is OmniLight3D.

RID **particles_collision_create**()

Creates a new 3D GPU particle collision or attractor and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID can be used in most `particles_collision_*` RenderingServer functions.

Note: The equivalent nodes are GPUParticlesCollision3D and GPUParticlesAttractor3D.

---

void **particles_collision_height_field_update**(particles_collision: RID)

Requests an update for the 3D GPU particle collision heightfield. This may be automatically called by the 3D GPU particle collision heightfield depending on its GPUParticlesCollisionHeightField3D.update_mode.

---

void **particles_collision_set_attractor_attenuation**(particles_collision: RID, curve: float)

Sets the attenuation `curve` for the 3D GPU particles attractor specified by the `particles_collision` RID. Only used for attractors, not colliders. Equivalent to GPUParticlesAttractor3D.attenuation.

---

void **particles_collision_set_attractor_directionality**(particles_collision: RID, amount: float)

Sets the directionality `amount` for the 3D GPU particles attractor specified by the `particles_collision` RID. Only used for attractors, not colliders. Equivalent to GPUParticlesAttractor3D.directionality.

---

void **particles_collision_set_attractor_strength**(particles_collision: RID, strength: float)

Sets the `strength` for the 3D GPU particles attractor specified by the `particles_collision` RID. Only used for attractors, not colliders. Equivalent to GPUParticlesAttractor3D.strength.

---

void **particles_collision_set_box_extents**(particles_collision: RID, extents: Vector3)

Sets the `extents` for the 3D GPU particles collision by the `particles_collision` RID. Equivalent to GPUParticlesCollisionBox3D.size, GPUParticlesCollisionSDF3D.size, GPUParticlesCollisionHeightField3D.size, GPUParticlesAttractorBox3D.size or GPUParticlesAttractorVectorField3D.size depending on the `particles_collision` type.

void **particles_collision_set_collision_type**(particles_collision: RID, type: ParticlesCollisionType)

Sets the collision or attractor shape `type` for the 3D GPU particles collision or attractor specified by the `particles_collision` RID.

void **particles_collision_set_cull_mask**(particles_collision: RID, mask: int)

Sets the cull `mask` for the 3D GPU particles collision or attractor specified by the `particles_collision` RID. Equivalent to GPUParticlesCollision3D.cull_mask or GPUParticlesAttractor3D.cull_mask depending on the `particles_collision` type.

void **particles_collision_set_field_texture**(particles_collision: RID, texture: RID)

Sets the signed distance field `texture` for the 3D GPU particles collision specified by the `particles_collision` RID. Equivalent to GPUParticlesCollisionSDF3D.texture or GPUParticlesAttractorVectorField3D.texture depending on the `particles_collision` type.

void **particles_collision_set_height_field_resolution**(particles_collision: RID, resolution: ParticlesCollisionHeightfieldResolution)

Sets the heightmap `resolution` for the 3D GPU particles heightfield collision specified by the `particles_collision` RID. Equivalent to GPUParticlesCollisionHeightField3D.resolution.

void **particles_collision_set_sphere_radius**(particles_collision: RID, radius: float)

Sets the `radius` for the 3D GPU particles sphere collision or attractor specified by the `particles_collision` RID. Equivalent to GPUParticlesCollisionSphere3D.radius or GPUParticlesAttractorSphere3D.radius depending on the `particles_collision` type.

RID **particles_create**()

Creates a GPU-based particle system and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `particles_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

To place in a scene, attach these particles to an instance using instance_set_base using the returned RID.

Note: The equivalent nodes are GPUParticles2D and GPUParticles3D.

Note: All `particles_*` methods only apply to GPU-based particles, not CPU-based particles. CPUParticles2D and CPUParticles3D do not have equivalent RenderingServer functions available, as these use MultiMeshInstance2D and MultiMeshInstance3D under the hood (see `multimesh_*` methods).

void **particles_emit**(particles: RID, transform: Transform3D, velocity: Vector3, color: Color, custom: Color, emit_flags: int)

Manually emits particles from the `particles` instance.

AABB **particles_get_current_aabb**(particles: RID)

Calculates and returns the axis-aligned bounding box that contains all the particles. Equivalent to GPUParticles3D.capture_aabb.

bool **particles_get_emitting**(particles: RID)

Returns `true` if particles are currently set to emitting.

bool **particles_is_inactive**(particles: RID)

Returns `true` if particles are not emitting and particles are set to inactive.

void **particles_request_process**(particles: RID)

Add particle system to list of particle systems that need to be updated. Update will take place on the next frame, or on the next call to instances_cull_aabb, instances_cull_convex, or instances_cull_ray.

void **particles_restart**(particles: RID)

Reset the particles on the next update. Equivalent to GPUParticles3D.restart.

void **particles_set_amount**(particles: RID, amount: int)

Sets the number of particles to be drawn and allocates the memory for them. Equivalent to GPUParticles3D.amount.

void **particles_set_amount_ratio**(particles: RID, ratio: float)

Sets the amount ratio for particles to be emitted. Equivalent to GPUParticles3D.amount_ratio.

void **particles_set_collision_base_size**(particles: RID, size: float)

There is currently no description for this method. Please help us by contributing one!

void **particles_set_custom_aabb**(particles: RID, aabb: AABB) 🔗

Sets a custom axis-aligned bounding box for the particle system. Equivalent to GPUParticles3D.visibility_aabb.

---

void **particles_set_draw_order**(particles: RID, order: ParticlesDrawOrder) 🔗

Sets the draw order of the particles to one of the named enums from ParticlesDrawOrder. See ParticlesDrawOrder for options. Equivalent to GPUParticles3D.draw_order.

---

void **particles_set_draw_pass_mesh**(particles: RID, pass: int, mesh: RID) 🔗

Sets the mesh to be used for the specified draw pass. Equivalent to GPUParticles3D.draw_pass_1, GPUParticles3D.draw_pass_2, GPUParticles3D.draw_pass_3, and GPUParticles3D.draw_pass_4.

---

void **particles_set_draw_passes**(particles: RID, count: int) 🔗

Sets the number of draw passes to use. Equivalent to GPUParticles3D.draw_passes.

---

void **particles_set_emission_transform**(particles: RID, transform: Transform3D) 🔗

Sets the Transform3D that will be used by the particles when they first emit.

---

void **particles_set_emitter_velocity**(particles: RID, velocity: Vector3) 🔗

Sets the velocity of a particle node, that will be used by ParticleProcessMaterial.inherit_velocity_ratio.

---

void **particles_set_emitting**(particles: RID, emitting: bool) 🔗

If `true`, particles will emit over time. Setting to false does not reset the particles, but only stops their emission. Equivalent to GPUParticles3D.emitting.

---

void **particles_set_explosiveness_ratio**(particles: RID, ratio: float) 🔗

Sets the explosiveness ratio. Equivalent to GPUParticles3D.explosiveness.

---

void **particles_set_fixed_fps**(particles: RID, fps: int) 🔗

Sets the frame rate that the particle system rendering will be fixed to. Equivalent to GPUParticles3D.fixed_fps.

---

void **particles_set_fractional_delta**(particles: RID, enable: bool) 🔗

If `true`, uses fractional delta which smooths the movement of the particles. Equivalent to GPUParticles3D.fract_delta.

---

void **particles_set_interp_to_end**(particles: RID, factor: float)

Sets the value that informs a ParticleProcessMaterial to rush all particles towards the end of their lifetime.

---

void **particles_set_interpolate**(particles: RID, enable: bool)

There is currently no description for this method. Please help us by contributing one!

---

void **particles_set_lifetime**(particles: RID, lifetime: float)

Sets the lifetime of each particle in the system. Equivalent to GPUParticles3D.lifetime.

---

void **particles_set_mode**(particles: RID, mode: ParticlesMode)

Sets whether the GPU particles specified by the `particles` RID should be rendered in 2D or 3D according to `mode`.

---

void **particles_set_one_shot**(particles: RID, one_shot: bool)

If `true`, particles will emit once and then stop. Equivalent to GPUParticles3D.one_shot.

---

void **particles_set_pre_process_time**(particles: RID, time: float)

Sets the preprocess time for the particles' animation. This lets you delay starting an animation until after the particles have begun emitting. Equivalent to GPUParticles3D.preprocess.

---

void **particles_set_process_material**(particles: RID, material: RID)

Sets the material for processing the particles.

Note: This is not the material used to draw the materials. Equivalent to GPUParticles3D.process_material.

---

void **particles_set_randomness_ratio**(particles: RID, ratio: float)

Sets the emission randomness ratio. This randomizes the emission of particles within their phase. Equivalent to GPUParticles3D.randomness.

---

void **particles_set_speed_scale**(particles: RID, scale: float)

Sets the speed scale of the particle system. Equivalent to GPUParticles3D.speed_scale.

---

void **particles_set_subemitter**(particles: RID, subemitter_particles: RID)

There is currently no description for this method. Please help us by contributing one!

---

void **particles_set_trail_bind_poses**(particles: RID, bind_poses: Array[Transform3D]) 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **particles_set_trails**(particles: RID, enable: bool, length_sec: float) 🔗

If `enable` is `true`, enables trails for the `particles` with the specified `length_sec` in seconds. Equivalent to GPUParticles3D.trail_enabled and GPUParticles3D.trail_lifetime.

---

void **particles_set_transform_align**(particles: RID, align: ParticlesTransformAlign) 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **particles_set_use_local_coordinates**(particles: RID, enable: bool) 🔗

If `true`, particles use local coordinates. If `false` they use global coordinates. Equivalent to GPUParticles3D.local_coords.

---

void **positional_soft_shadow_filter_set_quality**(quality: ShadowQuality) 🔗

Sets the filter quality for omni and spot light shadows in 3D. See also ProjectSettings.rendering/lights_and_shadows/positional_shadow/soft_shadow_filter_quality. This parameter is global and cannot be set on a per-viewport basis.

RID **reflection_probe_create**() 🔗

Creates a reflection probe and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `reflection_probe_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

To place in a scene, attach this reflection probe to an instance using instance_set_base using the returned RID.

Note: The equivalent node is ReflectionProbe.

---

void **reflection_probe_set_ambient_color**(probe: RID, color: Color) 🔗

Sets the reflection probe's custom ambient light color. Equivalent to ReflectionProbe.ambient_color.

---

void **reflection_probe_set_ambient_energy**(probe: RID, energy: float) 🔗

Sets the reflection probe's custom ambient light energy. Equivalent to ReflectionProbe.ambient_color_energy.

void **reflection_probe_set_ambient_mode**(probe: RID, mode: ReflectionProbeAmbientMode)

Sets the reflection probe's ambient light mode. Equivalent to ReflectionProbe.ambient_mode.

void **reflection_probe_set_as_interior**(probe: RID, enable: bool)

If `true`, reflections will ignore sky contribution. Equivalent to ReflectionProbe.interior.

void **reflection_probe_set_cull_mask**(probe: RID, layers: int)

Sets the render cull mask for this reflection probe. Only instances with a matching layer will be reflected by this probe. Equivalent to ReflectionProbe.cull_mask.

void **reflection_probe_set_enable_box_projection**(probe: RID, enable: bool)

If `true`, uses box projection. This can make reflections look more correct in certain situations. Equivalent to ReflectionProbe.box_projection.

void **reflection_probe_set_enable_shadows**(probe: RID, enable: bool)

If `true`, computes shadows in the reflection probe. This makes the reflection much slower to compute. Equivalent to ReflectionProbe.enable_shadows.

void **reflection_probe_set_intensity**(probe: RID, intensity: float)

Sets the intensity of the reflection probe. Intensity modulates the strength of the reflection. Equivalent to ReflectionProbe.intensity.

void **reflection_probe_set_max_distance**(probe: RID, distance: float)

Sets the max distance away from the probe an object can be before it is culled. Equivalent to ReflectionProbe.max_distance.

void **reflection_probe_set_mesh_lod_threshold**(probe: RID, pixels: float)

Sets the mesh level of detail to use in the reflection probe rendering. Higher values will use less detailed versions of meshes that have LOD variations generated, which can improve performance. Equivalent to ReflectionProbe.mesh_lod_threshold.

void **reflection_probe_set_origin_offset**(probe: RID, offset: Vector3)

Sets the origin offset to be used when this reflection probe is in box project mode. Equivalent to ReflectionProbe.origin_offset.

---

void **reflection_probe_set_reflection_mask**(probe: RID, layers: int)

Sets the render reflection mask for this reflection probe. Only instances with a matching layer will have reflections applied from this probe. Equivalent to ReflectionProbe.reflection_mask.

---

void **reflection_probe_set_resolution**(probe: RID, resolution: int)

Sets the resolution to use when rendering the specified reflection probe. The `resolution` is specified for each cubemap face: for instance, specifying `512` will allocate 6 faces of 512×512 each (plus mipmaps for roughness levels).

---

void **reflection_probe_set_size**(probe: RID, size: Vector3)

Sets the size of the area that the reflection probe will capture. Equivalent to ReflectionProbe.size.

---

void **reflection_probe_set_update_mode**(probe: RID, mode: ReflectionProbeUpdateMode)

Sets how often the reflection probe updates. Can either be once or every frame. See ReflectionProbeUpdateMode for options.

---

void **request_frame_drawn_callback**(callable: Callable)

Schedules a callback to the given callable after a frame has been drawn.

RID **scenario_create**()

Creates a scenario and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `scenario_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

The scenario is the 3D world that all the visual instances exist in.

---

void **scenario_set_camera_attributes**(scenario: RID, effects: RID)

Sets the camera attributes (`effects`) that will be used with this scenario. See also CameraAttributes.

---

void **scenario_set_compositor**(scenario: RID, compositor: RID)

Sets the compositor (`compositor`) that will be used with this scenario. See also Compositor.

---

void **scenario_set_environment**(scenario: RID, environment: RID)

Sets the environment that will be used with this scenario. See also Environment.

---

void **scenario_set_fallback_environment**(scenario: RID, environment: RID)

Sets the fallback environment to be used by this scenario. The fallback environment is used if no environment is set. Internally, this is used by the editor to provide a default environment.

---

void **screen_space_roughness_limiter_set_active**(enable: bool, amount: float, limit: float)

Sets the screen-space roughness limiter parameters, such as whether it should be enabled and its thresholds. Equivalent to ProjectSettings.rendering/anti_aliasing/screen_space_roughness_limiter/enabled, ProjectSettings.rendering/anti_aliasing/screen_space_roughness_limiter/amount and ProjectSettings.rendering/anti_aliasing/screen_space_roughness_limiter/limit.

---

void **set_boot_image**(image: Image, color: Color, scale: bool, use_filter: bool = true)

Sets a boot image. The color defines the background color. If `scale` is `true`, the image will be scaled to fit the screen size. If `use_filter` is `true`, the image will be scaled with linear interpolation. If `use_filter` is `false`, the image will be scaled with nearest-neighbor interpolation.

---

void **set_debug_generate_wireframes**(generate: bool)

This method is currently unimplemented and does nothing if called with `generate` set to `true`.

---

void **set_default_clear_color**(color: Color)

Sets the default clear color which is used when a specific clear color has not been selected. See also get_default_clear_color.

RID **shader_create**()

Creates an empty shader and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `shader_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is Shader.

String **shader_get_code**(shader: RID) const

Returns a shader's source code as a string.

RID **shader_get_default_texture_parameter**(shader: RID, name: StringName, index: int = 0) const

Returns a default texture from a shader searched by name.

Note: If the sampler array is used use `index` to access the specified texture.

Variant **shader_get_parameter_default**(shader: RID, name: StringName) const

Returns the default value for the specified shader uniform. This is usually the value written in the shader source code.

void **shader_set_code**(shader: RID, code: String)

Sets the shader's source code (which triggers recompilation after being changed).

void **shader_set_default_texture_parameter**(shader: RID, name: StringName, texture: RID, index: int = 0)

Sets a shader's default texture. Overwrites the texture given by name.

Note: If the sampler array is used use `index` to access the specified texture.

void **shader_set_path_hint**(shader: RID, path: String)

Sets the path hint for the specified shader. This should generally match the Shader resource's Resource.resource_path.

void **skeleton_allocate_data**(skeleton: RID, bones: int, is_2d_skeleton: bool = false)

There is currently no description for this method. Please help us by contributing one!

Transform3D **skeleton_bone_get_transform**(skeleton: RID, bone: int) const

Returns the Transform3D set for a specific bone of this skeleton.

Transform2D **skeleton_bone_get_transform_2d**(skeleton: RID, bone: int) const

Returns the Transform2D set for a specific bone of this skeleton.

void **skeleton_bone_set_transform**(skeleton: RID, bone: int, transform: Transform3D)

Sets the Transform3D for a specific bone of this skeleton.

void **skeleton_bone_set_transform_2d**(skeleton: RID, bone: int, transform: Transform2D)

Sets the Transform2D for a specific bone of this skeleton.

RID **skeleton_create**()

Creates a skeleton and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `skeleton_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

---

int **skeleton_get_bone_count**(skeleton: RID) const

Returns the number of bones allocated for this skeleton.

---

void **skeleton_set_base_transform_2d**(skeleton: RID, base_transform: Transform2D)

There is currently no description for this method. Please help us by contributing one!

Image **sky_bake_panorama**(sky: RID, energy: float, bake_irradiance: bool, size: Vector2i)

Generates and returns an Image containing the radiance map for the specified `sky` RID. This supports built-in sky material and custom sky shaders. If `bake_irradiance` is `true`, the irradiance map is saved instead of the radiance map. The radiance map is used to render reflected light, while the irradiance map is used to render ambient light. See also environment_bake_panorama.

Note: The image is saved in linear color space without any tonemapping performed, which means it will look too dark if viewed directly in an image editor. `energy` values above `1.0` can be used to brighten the resulting image.

Note: `size` should be a 2:1 aspect ratio for the generated panorama to have square pixels. For radiance maps, there is no point in using a height greater than Sky.radiance_size, as it won't increase detail. Irradiance maps only contain low-frequency data, so there is usually no point in going past a size of 128×64 pixels when saving an irradiance map.

RID **sky_create**()

Creates an empty sky and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `sky_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

---

void **sky_set_material**(sky: RID, material: RID)

Sets the material that the sky uses to render the background, ambient and reflection maps.

---

void **sky_set_mode**(sky: RID, mode: SkyMode)

Sets the process `mode` of the sky specified by the `sky` RID. Equivalent to Sky.process_mode.

---

void **sky_set_radiance_size**(sky: RID, radiance_size: int) 🔗

Sets the `radiance_size` of the sky specified by the `sky` RID (in pixels). Equivalent to Sky.radiance_size.

RID **spot_light_create**() 🔗

Creates a spot light and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID can be used in most `light_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

To place in a scene, attach this spot light to an instance using instance_set_base using the returned RID.

---

void **sub_surface_scattering_set_quality**(quality: SubSurfaceScatteringQuality) 🔗

Sets ProjectSettings.rendering/environment/subsurface_scattering/subsurface_scattering_quality to use when rendering materials that have subsurface scattering enabled.

---

void **sub_surface_scattering_set_scale**(scale: float, depth_scale: float) 🔗

Sets the ProjectSettings.rendering/environment/subsurface_scattering/subsurface_scattering_scale and ProjectSettings.rendering/environment/subsurface_scattering/subsurface_scattering_depth_scale to use when rendering materials that have subsurface scattering enabled.

RID **texture_2d_create**(image: Image) 🔗

Creates a 2-dimensional texture and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `texture_2d_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is Texture2D.

Note: Not to be confused with RenderingDevice.texture_create, which creates the graphics API's own texture type as opposed to the Godot-specific Texture2D resource.

Image **texture_2d_get**(texture: RID) const 🔗

Returns an Image instance from the given `texture` RID.

Example of getting the test texture from get_test_texture and applying it to a Sprite2D node:

```
var texture_rid = RenderingServer.get_test_texture()
var texture = ImageTexture.create_from_image(RenderingServer.texture_2d_get(texture_rid))
$Sprite2D.texture = texture
```

Image **texture_2d_layer_get**(texture: RID, layer: int) const

Returns an Image instance from the given `texture` RID and `layer`.

RID **texture_2d_layered_create**(layers: Array[Image], layered_type: TextureLayeredType)

Creates a 2-dimensional layered texture and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `texture_2d_layered_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is TextureLayered.

RID **texture_2d_layered_placeholder_create**(layered_type: TextureLayeredType)

Creates a placeholder for a 2-dimensional layered texture and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `texture_2d_layered_*` RenderingServer functions, although it does nothing when used. See also texture_2d_placeholder_create.

Note: The equivalent resource is PlaceholderTextureLayered.

RID **texture_2d_placeholder_create**()

Creates a placeholder for a 2-dimensional layered texture and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `texture_2d_layered_*` RenderingServer functions, although it does nothing when used. See also texture_2d_layered_placeholder_create

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is PlaceholderTexture2D.

---

void **texture_2d_update**(texture: RID, image: Image, layer: int)

Updates the texture specified by the `texture` RID with the data in `image`. A `layer` must also be specified, which should be `0` when updating a single-layer texture (Texture2D).

Note: The `image` must have the same width, height and format as the current `texture` data. Otherwise, an error will be printed and the original texture won't be modified. If you need to use different width, height or format, use texture_replace instead.

RID **texture_3d_create**(format: Format, width: int, height: int, depth: int, mipmaps: bool, data: Array[Image]) 🔗

Note: The equivalent resource is Texture3D.

Array[Image] **texture_3d_get**(texture: RID) const 🔗

Returns 3D texture data as an array of Images for the specified texture RID.

RID **texture_3d_placeholder_create**() 🔗

Creates a placeholder for a 3-dimensional texture and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `texture_3d_*` RenderingServer functions, although it does nothing when used.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent resource is PlaceholderTexture3D.

---

void **texture_3d_update**(texture: RID, data: Array[Image]) 🔗

Updates the texture specified by the `texture` RID's data with the data in `data`. All the texture's layers must be replaced at once.

Note: The `texture` must have the same width, height, depth and format as the current texture data. Otherwise, an error will be printed and the original texture won't be modified. If you need to use different width, height, depth or format, use texture_replace instead.

Format **texture_get_format**(texture: RID) const 🔗

Returns the format for the texture.

---

int **texture_get_native_handle**(texture: RID, srgb: bool = false) const 🔗

Returns the internal graphics handle for this texture object. For use when communicating with third-party APIs mostly with GDExtension.

Note: This function returns a `uint64_t` which internally maps to a `GLuint` (OpenGL) or `VkImage` (Vulkan).

String **texture_get_path**(texture: RID) const 🔗

There is currently no description for this method. Please help us by contributing one!

RID **texture_get_rd_texture**(texture: RID, srgb: bool = false) const 🔗

Returns a texture RID that can be used with RenderingDevice.

RID **texture_proxy_create**(base: RID)

Deprecated: ProxyTexture was removed in Godot 4.

This method does nothing and always returns an invalid RID.

---

void **texture_proxy_update**(texture: RID, proxy_to: RID)

Deprecated: ProxyTexture was removed in Godot 4.

This method does nothing.

RID **texture_rd_create**(rd_texture: RID, layer_type: TextureLayeredType = 0)

Creates a new texture object based on a texture created directly on the RenderingDevice. If the texture contains layers, `layer_type` is used to define the layer type.

---

void **texture_replace**(texture: RID, by_texture: RID)

Replaces `texture`'s texture data by the texture specified by the `by_texture` RID, without changing `texture`'s RID.

---

void **texture_set_force_redraw_if_visible**(texture: RID, enable: bool)

There is currently no description for this method. Please help us by contributing one!

---

void **texture_set_path**(texture: RID, path: String)

There is currently no description for this method. Please help us by contributing one!

---

void **texture_set_size_override**(texture: RID, width: int, height: int)

There is currently no description for this method. Please help us by contributing one!

---

void **viewport_attach_camera**(viewport: RID, camera: RID)

Sets a viewport's camera.

---

void **viewport_attach_canvas**(viewport: RID, canvas: RID)

Sets a viewport's canvas.

---

void **viewport_attach_to_screen**(viewport: RID, rect: Rect2 = Rect2(0, 0, 0, 0), screen: int = 0) 🔗

Copies the viewport to a region of the screen specified by `rect`. If viewport_set_render_direct_to_screen is `true`, then the viewport does not use a framebuffer and the contents of the viewport are rendered directly to screen. However, note that the root viewport is drawn last, therefore it will draw over the screen. Accordingly, you must set the root viewport to an area that does not cover the area that you have attached this viewport to.

For example, you can set the root viewport to not render at all with the following code:

FIXME: The method seems to be non-existent.

GDScript

```
func _ready():
    get_viewport().set_attach_to_screen_rect(Rect2())
    $Viewport.set_attach_to_screen_rect(Rect2(0, 0, 600, 600))
```

Using this can result in significant optimization, especially on lower-end devices. However, it comes at the cost of having to manage your viewports manually. For further optimization, see viewport_set_render_direct_to_screen.

RID **viewport_create**() 🔗

Creates an empty viewport and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `viewport_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent node is Viewport.

---

float **viewport_get_measured_render_time_cpu**(viewport: RID) const 🔗

Returns the CPU time taken to render the last frame in milliseconds. This *only* includes time spent in rendering-related operations; scripts' `_process` functions and other engine subsystems are not included in this readout. To get a complete readout of CPU time spent to render the scene, sum the render times of all viewports that are drawn every frame plus get_frame_setup_time_cpu. Unlike Engine.get_frames_per_second, this method will accurately reflect CPU utilization even if framerate is capped via V-Sync or Engine.max_fps. See also viewport_get_measured_render_time_gpu.

Note: Requires measurements to be enabled on the specified `viewport` using viewport_set_measure_render_time. Otherwise, this method returns `0.0`.

---

float **viewport_get_measured_render_time_gpu**(viewport: RID) const 🔗

Returns the GPU time taken to render the last frame in milliseconds. To get a complete readout of GPU time spent to render the scene, sum the render times of all viewports that are drawn every frame. Unlike Engine.get_frames_per_second, this method accurately reflects GPU utilization even if framerate is capped via V-Sync or Engine.max_fps. See also viewport_get_measured_render_time_gpu.

Note: Requires measurements to be enabled on the specified `viewport` using viewport_set_measure_render_time. Otherwise, this method returns `0.0`.

Note: When GPU utilization is low enough during a certain period of time, GPUs will decrease their power state (which in turn decreases core and memory clock speeds). This can cause the reported GPU time to increase if GPU utilization is kept low enough by a framerate cap (compared to what it would be at the GPU's highest power state). Keep this in mind when benchmarking using viewport_get_measured_render_time_gpu. This behavior can be overridden in the graphics driver settings at the cost of higher power usage.

---

int **viewport_get_render_info**(viewport: RID, type: ViewportRenderInfoType, info: ViewportRenderInfo) 🔗

Returns a statistic about the rendering engine which can be used for performance profiling. This is separated into render pass `type`s, each of them having the same `info`s you can query (different passes will return different values). See ViewportRenderInfoType for a list of render pass types and ViewportRenderInfo for a list of information that can be queried.

See also get_rendering_info, which returns global information across all viewports.

Note: Viewport rendering information is not available until at least 2 frames have been rendered by the engine. If rendering information is not available, viewport_get_render_info returns `0`. To print rendering information in `_ready()` successfully, use the following:

```
func _ready():
    for _i in 2:
        await get_tree().process_frame

    print(
            RenderingServer.viewport_get_render_info(get_viewport().get_viewport_rid(),
            RenderingServer.VIEWPORT_RENDER_INFO_TYPE_VISIBLE,
            RenderingServer.VIEWPORT_RENDER_INFO_DRAW_CALLS_IN_FRAME)
    )
```

RID **viewport_get_render_target**(viewport: RID) const 🔗

Returns the render target for the viewport.

RID **viewport_get_texture**(viewport: RID) const 🔗

Returns the viewport's last rendered frame.

ViewportUpdateMode **viewport_get_update_mode**(viewport: RID) const 🔗

Returns the viewport's update mode. See ViewportUpdateMode constants for options.

Warning: Calling this from any thread other than the rendering thread will be detrimental to performance.

void **viewport_remove_canvas**(viewport: RID, canvas: RID)

Detaches a viewport from a canvas.

void **viewport_set_active**(viewport: RID, active: bool)

If `true`, sets the viewport active, else sets it inactive.

void **viewport_set_canvas_cull_mask**(viewport: RID, canvas_cull_mask: int)

Sets the rendering mask associated with this Viewport. Only CanvasItem nodes with a matching rendering visibility layer will be rendered by this Viewport.

void **viewport_set_canvas_stacking**(viewport: RID, canvas: RID, layer: int, sublayer: int)

Sets the stacking order for a viewport's canvas.

`layer` is the actual canvas layer, while `sublayer` specifies the stacking order of the canvas among those in the same layer.

void **viewport_set_canvas_transform**(viewport: RID, canvas: RID, offset: Transform2D)

Sets the transformation of a viewport's canvas.

void **viewport_set_clear_mode**(viewport: RID, clear_mode: ViewportClearMode)

Sets the clear mode of a viewport. See ViewportClearMode for options.

void **viewport_set_debug_draw**(viewport: RID, draw: ViewportDebugDraw)

Sets the debug draw mode of a viewport. See ViewportDebugDraw for options.

void **viewport_set_default_canvas_item_texture_filter**(viewport: RID, filter: CanvasItemTextureFilter)

Sets the default texture filtering mode for the specified `viewport` RID. See CanvasItemTextureFilter for options.

void **viewport_set_default_canvas_item_texture_repeat**(viewport: RID, repeat: CanvasItemTextureRepeat)

Sets the default texture repeat mode for the specified `viewport` RID. See CanvasItemTextureRepeat for options.

void **viewport_set_disable_2d**(viewport: RID, disable: bool)

If `true`, the viewport's canvas (i.e. 2D and GUI elements) is not rendered.

---

void **viewport_set_disable_3d**(viewport: RID, disable: bool)

If `true`, the viewport's 3D elements are not rendered.

---

void **viewport_set_environment_mode**(viewport: RID, mode: ViewportEnvironmentMode)

Sets the viewport's environment mode which allows enabling or disabling rendering of 3D environment over 2D canvas. When disabled, 2D will not be affected by the environment. When enabled, 2D will be affected by the environment if the environment background mode is ENV_BG_CANVAS. The default behavior is to inherit the setting from the viewport's parent. If the topmost parent is also set to VIEWPORT_ENVIRONMENT_INHERIT, then the behavior will be the same as if it was set to VIEWPORT_ENVIRONMENT_ENABLED.

---

void **viewport_set_fsr_sharpness**(viewport: RID, sharpness: float)

Determines how sharp the upscaled image will be when using the FSR upscaling mode. Sharpness halves with every whole number. Values go from 0.0 (sharpest) to 2.0. Values above 2.0 won't make a visible difference.

---

void **viewport_set_global_canvas_transform**(viewport: RID, transform: Transform2D)

Sets the viewport's global transformation matrix.

---

void **viewport_set_measure_render_time**(viewport: RID, enable: bool)

Sets the measurement for the given `viewport` RID (obtained using Viewport.get_viewport_rid). Once enabled, viewport_get_measured_render_time_cpu and viewport_get_measured_render_time_gpu will return values greater than `0.0` when queried with the given `viewport`.

---

void **viewport_set_msaa_2d**(viewport: RID, msaa: ViewportMSAA)

Sets the multisample anti-aliasing mode for 2D/Canvas on the specified `viewport` RID. See ViewportMSAA for options.

---

void **viewport_set_msaa_3d**(viewport: RID, msaa: ViewportMSAA)

Sets the multisample anti-aliasing mode for 3D on the specified `viewport` RID. See ViewportMSAA for options.

---

void **viewport_set_occlusion_culling_build_quality**(quality: ViewportOcclusionCullingBuildQuality)

Sets the ProjectSettings.rendering/occlusion_culling/bvh_build_quality to use for occlusion culling. This parameter is global and cannot be set on a per-viewport basis.

---

void **viewport_set_occlusion_rays_per_thread**(rays_per_thread: int)

Sets the ProjectSettings.rendering/occlusion_culling/occlusion_rays_per_thread to use for occlusion culling. This parameter is global and cannot be set on a per-viewport basis.

---

void **viewport_set_parent_viewport**(viewport: RID, parent_viewport: RID)

Sets the viewport's parent to the viewport specified by the `parent_viewport` RID.

---

void **viewport_set_positional_shadow_atlas_quadrant_subdivision**(viewport: RID, quadrant: int, subdivision: int)

Sets the number of subdivisions to use in the specified shadow atlas `quadrant` for omni and spot shadows. See also Viewport.set_positional_shadow_atlas_quadrant_subdiv.

---

void **viewport_set_positional_shadow_atlas_size**(viewport: RID, size: int, use_16_bits: bool = false)

Sets the `size` of the shadow atlas's images (used for omni and spot lights) on the viewport specified by the `viewport` RID. The value is rounded up to the nearest power of 2. If `use_16_bits` is `true`, use 16 bits for the omni/spot shadow depth map. Enabling this results in shadows having less precision and may result in shadow acne, but can lead to performance improvements on some devices.

Note: If this is set to `0`, no positional shadows will be visible at all. This can improve performance significantly on low-end systems by reducing both the CPU and GPU load (as fewer draw calls are needed to draw the scene without shadows).

---

void **viewport_set_render_direct_to_screen**(viewport: RID, enabled: bool)

If `true`, render the contents of the viewport directly to screen. This allows a low-level optimization where you can skip drawing a viewport to the root viewport. While this optimization can result in a significant increase in speed (especially on older devices), it comes at a cost of usability. When this is enabled, you cannot read from the viewport or from the screen_texture. You also lose the benefit of certain window settings, such as the various stretch modes. Another consequence to be aware of is that in 2D the rendering happens in window coordinates, so if you have a viewport that is double the size of the window, and you set this, then only the portion that fits within the window will be drawn, no automatic scaling is possible, even if your game scene is significantly larger than the window size.

---

void **viewport_set_scaling_3d_mode**(viewport: RID, scaling_3d_mode: ViewportScaling3DMode)

Sets the 3D resolution scaling mode. Bilinear scaling renders at different resolution to either undersample or supersample the viewport. FidelityFX Super Resolution 1.0, abbreviated to FSR, is an upscaling technology that produces high quality images at fast framerates by using a spatially aware upscaling algorithm. FSR is slightly more expensive than bilinear, but it produces significantly higher image quality. FSR should be used where possible.

---

void **viewport_set_scaling_3d_scale**(viewport: RID, scale: float)

Scales the 3D render buffer based on the viewport size uses an image filter specified in ViewportScaling3DMode to scale the output image to the full viewport size. Values lower than `1.0` can be used to speed up 3D rendering at the cost of quality (undersampling). Values greater than `1.0` are only valid for bilinear mode and can be used to improve 3D rendering quality at a high performance cost (supersampling). See also ViewportMSAA for multi-sample antialiasing, which is significantly cheaper but only smoothens the edges of polygons.

When using FSR upscaling, AMD recommends exposing the following values as preset options to users "Ultra Quality: 0.77", "Quality: 0.67", "Balanced: 0.59", "Performance: 0.5" instead of exposing the entire scale.

---

void **viewport_set_scenario**(viewport: RID, scenario: RID)

Sets a viewport's scenario. The scenario contains information about environment information, reflection atlas, etc.

---

void **viewport_set_screen_space_aa**(viewport: RID, mode: ViewportScreenSpaceAA)

Sets the viewport's screen-space antialiasing mode.

---

void **viewport_set_sdf_oversize_and_scale**(viewport: RID, oversize: ViewportSDFOversize, scale: ViewportSDFScale)

Sets the viewport's 2D signed distance field ProjectSettings.rendering/2d/sdf/oversize and ProjectSettings.rendering/2d/sdf/scale. This is used when sampling the signed distance field in CanvasItem shaders as well as GPUParticles2D collision. This is *not* used by SDFGI in 3D rendering.

---

void **viewport_set_size**(viewport: RID, width: int, height: int)

Sets the viewport's width and height in pixels.

---

void **viewport_set_snap_2d_transforms_to_pixel**(viewport: RID, enabled: bool)

If `true`, canvas item transforms (i.e. origin position) are snapped to the nearest pixel when rendering. This can lead to a crisper appearance at the cost of less smooth movement, especially when Camera2D smoothing is enabled. Equivalent to ProjectSettings.rendering/2d/snap/snap_2d_transforms_to_pixel.

---

void **viewport_set_snap_2d_vertices_to_pixel**(viewport: RID, enabled: bool)

If `true`, canvas item vertices (i.e. polygon points) are snapped to the nearest pixel when rendering. This can lead to a crisper appearance at the cost of less smooth movement, especially when Camera2D smoothing is enabled. Equivalent to ProjectSettings.rendering/2d/snap/snap_2d_vertices_to_pixel.

---

void **viewport_set_texture_mipmap_bias**(viewport: RID, mipmap_bias: float)

Affects the final texture sharpness by reading from a lower or higher mipmap (also called "texture LOD bias"). Negative values make mipmapped textures sharper but grainier when viewed at a distance, while positive values make mipmapped textures blurrier (even when up close). To get sharper textures at a distance without introducing too much graininess, set this between `-0.75` and `0.0`. Enabling temporal antialiasing (ProjectSettings.rendering/anti_aliasing/quality/use_taa) can help reduce the graininess visible when using negative mipmap bias.

Note: When the 3D scaling mode is set to FSR 1.0, this value is used to adjust the automatic mipmap bias which is calculated internally based on the scale factor. The formula for this is `-log2(1.0 / scale) + mipmap_bias`.

---

void **viewport_set_transparent_background**(viewport: RID, enabled: bool)

If `true`, the viewport renders its background as transparent.

---

void **viewport_set_update_mode**(viewport: RID, update_mode: ViewportUpdateMode)

Sets when the viewport should be updated. See ViewportUpdateMode constants for options.

---

void **viewport_set_use_debanding**(viewport: RID, enable: bool)

If `true`, enables debanding on the specified viewport. Equivalent to ProjectSettings.rendering/anti_aliasing/quality/use_debanding.

---

void **viewport_set_use_hdr_2d**(viewport: RID, enabled: bool)

If `true`, 2D rendering will use a high dynamic range (HDR) format framebuffer matching the bit depth of the 3D framebuffer. When using the Forward+ renderer this will be an `RGBA16` framebuffer, while when using the Mobile renderer it will be an `RGB10_A2` framebuffer. Additionally, 2D rendering will take place in linear color space and will be converted to sRGB space immediately before blitting to the screen (if the Viewport is attached to the screen). Practically speaking, this means that the end result of the Viewport will not be clamped into the `0-1` range and can be used in 3D rendering without color space adjustments. This allows 2D rendering to take advantage of effects requiring high dynamic range (e.g. 2D glow) as well as substantially improves the appearance of effects requiring highly detailed gradients. This setting has the same effect as Viewport.use_hdr_2d.

Note: This setting will have no effect when using the GL Compatibility renderer as the GL Compatibility renderer always renders in low dynamic range for performance reasons.

---

void **viewport_set_use_occlusion_culling**(viewport: RID, enable: bool)

If `true`, enables occlusion culling on the specified viewport. Equivalent to ProjectSettings.rendering/occlusion_culling/use_occlusion_culling.

---

void **viewport_set_use_taa**(viewport: RID, enable: bool)

If `true`, use Temporal Anti-Aliasing. Equivalent to ProjectSettings.rendering/anti_aliasing/quality/use_taa.

---

void **viewport_set_use_xr**(viewport: RID, use_xr: bool)

If `true`, the viewport uses augmented or virtual reality technologies. See XRInterface.

---

void **viewport_set_vrs_mode**(viewport: RID, mode: ViewportVRSMode)

Sets the Variable Rate Shading (VRS) mode for the viewport. If the GPU does not support VRS, this property is ignored. Equivalent to ProjectSettings.rendering/vrs/mode.

---

void **viewport_set_vrs_texture**(viewport: RID, texture: RID)

The texture to use when the VRS mode is set to VIEWPORT_VRS_TEXTURE. Equivalent to ProjectSettings.rendering/vrs/texture.

---

void **viewport_set_vrs_update_mode**(viewport: RID, mode: ViewportVRSUpdateMode)

Sets the update mode for Variable Rate Shading (VRS) for the viewport. VRS requires the input texture to be converted to the format usable by the VRS method supported by the hardware. The update mode defines how often this happens. If the GPU does not support VRS, or VRS is not enabled, this property is ignored.

If set to VIEWPORT_VRS_UPDATE_ONCE, the input texture is copied once and the mode is changed to VIEWPORT_VRS_UPDATE_DISABLED.

RID **visibility_notifier_create**()

Creates a new 3D visibility notifier object and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `visibility_notifier_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

To place in a scene, attach this mesh to an instance using instance_set_base using the returned RID.

Note: The equivalent node is VisibleOnScreenNotifier3D.

---

void **visibility_notifier_set_aabb**(notifier: RID, aabb: AABB)

There is currently no description for this method. Please help us by contributing one!

---

void **visibility_notifier_set_callbacks**(notifier: RID, enter_callable: Callable, exit_callable: Callable)

There is currently no description for this method. Please help us by contributing one!

---

void **voxel_gi_allocate_data**(voxel_gi: RID, to_cell_xform: Transform3D, aabb: AABB, octree_size: Vector3i, octree_cells: PackedByteArray, data_cells: PackedByteArray, distance_field: PackedByteArray, level_counts: PackedInt32Array)

There is currently no description for this method. Please help us by contributing one!

RID **voxel_gi_create**()

Creates a new voxel-based global illumination object and adds it to the RenderingServer. It can be accessed with the RID that is returned. This RID will be used in all `voxel_gi_*` RenderingServer functions.

Once finished with your RID, you will want to free the RID using the RenderingServer's free_rid method.

Note: The equivalent node is VoxelGI.

PackedByteArray **voxel_gi_get_data_cells**(voxel_gi: RID) const

There is currently no description for this method. Please help us by contributing one!

PackedByteArray **voxel_gi_get_distance_field**(voxel_gi: RID) const

There is currently no description for this method. Please help us by contributing one!

PackedInt32Array **voxel_gi_get_level_counts**(voxel_gi: RID) const

There is currently no description for this method. Please help us by contributing one!

PackedByteArray **voxel_gi_get_octree_cells**(voxel_gi: RID) const

There is currently no description for this method. Please help us by contributing one!

Vector3i **voxel_gi_get_octree_size**(voxel_gi: RID) const

There is currently no description for this method. Please help us by contributing one!

Transform3D **voxel_gi_get_to_cell_xform**(voxel_gi: RID) const

There is currently no description for this method. Please help us by contributing one!

---

void **voxel_gi_set_baked_exposure_normalization**(voxel_gi: RID, baked_exposure: float) 🔗

Used to inform the renderer what exposure normalization value was used while baking the voxel gi. This value will be used and modulated at run time to ensure that the voxel gi maintains a consistent level of exposure even if the scene-wide exposure normalization is changed at run time. For more information see camera_attributes_set_exposure.

---

void **voxel_gi_set_bias**(voxel_gi: RID, bias: float) 🔗

Sets the VoxelGIData.bias value to use on the specified `voxel_gi`'s RID.

---

void **voxel_gi_set_dynamic_range**(voxel_gi: RID, range: float) 🔗

Sets the VoxelGIData.dynamic_range value to use on the specified `voxel_gi`'s RID.

---

void **voxel_gi_set_energy**(voxel_gi: RID, energy: float) 🔗

Sets the VoxelGIData.energy value to use on the specified `voxel_gi`'s RID.

---

void **voxel_gi_set_interior**(voxel_gi: RID, enable: bool) 🔗

Sets the VoxelGIData.interior value to use on the specified `voxel_gi`'s RID.

---

void **voxel_gi_set_normal_bias**(voxel_gi: RID, bias: float) 🔗

Sets the VoxelGIData.normal_bias value to use on the specified `voxel_gi`'s RID.

---

void **voxel_gi_set_propagation**(voxel_gi: RID, amount: float) 🔗

Sets the VoxelGIData.propagation value to use on the specified `voxel_gi`'s RID.

---

void **voxel_gi_set_quality**(quality: VoxelGIQuality) 🔗

Sets the ProjectSettings.rendering/global_illumination/voxel_gi/quality value to use when rendering. This parameter is global and cannot be set on a per-VoxelGI basis.

---

void **voxel_gi_set_use_two_bounces**(voxel_gi: RID, enable: bool) 🔗

Sets the VoxelGIData.use_two_bounces value to use on the specified `voxel_gi`'s RID.

RenderSceneBuffers

Inherits: RefCounted **<** Object

Inherited By: RenderSceneBuffersExtension, RenderSceneBuffersRD

Abstract scene buffers object, created for each viewport for which 3D rendering is done.

Description

Abstract scene buffers object, created for each viewport for which 3D rendering is done. It manages any additional buffers used during rendering and will discard buffers when the viewport is resized.

Note: This is an internal rendering server object, do not instantiate this from script.

Methods

void	configure(config: RenderSceneBuffersConfiguration)

Method Descriptions

void **configure**(config: RenderSceneBuffersConfiguration) 🔗

This method is called by the rendering server when the associated viewports configuration is changed. It will discard the old buffers and recreate the internal buffers used.

RenderSceneBuffersConfiguration

Inherits: RefCounted **<** Object

Configuration object used to setup a RenderSceneBuffers object.

Description

This configuration object is created and populated by the render engine on a viewport change and used to (re)configure a RenderSceneBuffers object.

Properties

float	fsr_sharpness	`0.0`
Vector2i	internal_size	`Vector2i(0, 0)`
ViewportMSAA	msaa_3d	`0`
RID	render_target	`RID()`
ViewportScaling3DMode	scaling_3d_mode	`255`

ViewportScreenSpaceAA	screen_space_aa	`0`
Vector2i	target_size	`Vector2i(0, 0)`
float	texture_mipmap_bias	`0.0`
int	view_count	`1`

Property Descriptions

float **fsr_sharpness** = `0.0` 🔗

- void **set_fsr_sharpness**(value: float)
- float **get_fsr_sharpness**()

FSR Sharpness applicable if FSR upscaling is used.

Vector2i **internal_size** = `Vector2i(0, 0)` 🔗

- void **set_internal_size**(value: Vector2i)
- Vector2i **get_internal_size**()

The size of the 3D render buffer used for rendering.

ViewportMSAA **msaa_3d** = `0` 🔗

- void **set_msaa_3d**(value: ViewportMSAA)
- ViewportMSAA **get_msaa_3d**()

The MSAA mode we're using for 3D rendering.

RID **render_target** = `RID()` 🔗

- void **set_render_target**(value: RID)
- RID **get_render_target**()

The render target associated with these buffer.

ViewportScaling3DMode **scaling_3d_mode** = `255` 🔗

- void **set_scaling_3d_mode**(value: ViewportScaling3DMode)
- ViewportScaling3DMode **get_scaling_3d_mode**()

The requested scaling mode with which we upscale/downscale if internal_size and target_size are not equal.

ViewportScreenSpaceAA **screen_space_aa** = `0` 🔗

- void **set_screen_space_aa**(value: ViewportScreenSpaceAA)
- ViewportScreenSpaceAA **get_screen_space_aa**()

The requested screen space AA applied in post processing.

Vector2i **target_size** = `Vector2i(0, 0)` 🔗

- void **set_target_size**(value: Vector2i)
- Vector2i **get_target_size**()

The target (upscale) size if scaling is used.

---

float **texture_mipmap_bias** = `0.0` 🔗

- void **set_texture_mipmap_bias**(value: float)
- float **get_texture_mipmap_bias**()

Bias applied to mipmaps.

---

int **view_count** = `1` 🔗

- void **set_view_count**(value: int)
- int **get_view_count**()

The number of views we're rendering.

RenderSceneBuffersExtension

Inherits: RenderSceneBuffers **<** RefCounted **<** Object

This class allows for a RenderSceneBuffer implementation to be made in GDExtension.

Description

This class allows for a RenderSceneBuffer implementation to be made in GDExtension.

Methods

void	_configure(config: RenderSceneBuffersConfiguration) virtual
void	_set_fsr_sharpness(fsr_sharpness: float) virtual
void	_set_texture_mipmap_bias(texture_mipmap_bias: float) virtual
void	_set_use_debanding(use_debanding: bool) virtual

Method Descriptions

void **_configure**(config: RenderSceneBuffersConfiguration) virtual

Implement this in GDExtension to handle the (re)sizing of a viewport.

void **_set_fsr_sharpness**(fsr_sharpness: float) virtual

Implement this in GDExtension to record a new FSR sharpness value.

void **_set_texture_mipmap_bias**(texture_mipmap_bias: float) virtual

Implement this in GDExtension to change the texture mipmap bias.

void **_set_use_debanding**(use_debanding: bool) virtual

Implement this in GDExtension to react to the debanding flag changing.

RenderSceneBuffersRD

Inherits: RenderSceneBuffers **<** RefCounted **<** Object

Render scene buffer implementation for the RenderingDevice based renderers.

Description

This object manages all 3D rendering buffers for the rendering device based renderers. An instance of this object is created for every viewport that has 3D rendering enabled.

All buffers are organized in **contexts**. The default context is called **render_buffers** and can contain amongst others the color buffer, depth buffer, velocity buffers, VRS density map and MSAA variants of these buffers.

Buffers are only guaranteed to exist during rendering of the viewport.

Note: This is an internal rendering server object, do not instantiate this from script.

Methods

void	clear_context(context: StringName)
RID	create_texture(context: StringName, name: StringName, data_format:

	DataFormat, usage_bits: int, texture_samples: TextureSamples, size: Vector2i, layers: int, mipmaps: int, unique: bool)
RID	create_texture_from_format(context: StringName, name: StringName, format: RDTextureFormat, view: RDTextureView, unique: bool)
RID	create_texture_view(context: StringName, name: StringName, view_name: StringName, view: RDTextureView)
RID	get_color_layer(layer: int, msaa: bool = false)
RID	get_color_texture(msaa: bool = false)
RID	get_depth_layer(layer: int, msaa: bool = false)
RID	get_depth_texture(msaa: bool = false)
float	get_fsr_sharpness() const
Vector2i	get_internal_size() const

ViewportMSAA	get_msaa_3d() const
RID	get_render_target() const
ViewportScaling3DMode	get_scaling_3d_mode() const
ViewportScreenSpaceAA	get_screen_space_aa() const
Vector2i	get_target_size() const
RID	get_texture(context: StringName, name: StringName) const
RDTextureFormat	get_texture_format(context: StringName, name: StringName) const
TextureSamples	get_texture_samples() const
RID	get_texture_slice(context: StringName, name: StringName, layer: int, mipmap: int, layers: int, mipmaps: int)
Vector2i	get_texture_slice_size(context: StringName, name: StringName, mipmap: int)

RID	get_texture_slice_view(context: StringName, name: StringName, layer: int, mipmap: int, layers: int, mipmaps: int, view: RDTextureView)
bool	get_use_debanding() const
bool	get_use_taa() const
RID	get_velocity_layer(layer: int, msaa: bool = false)
RID	get_velocity_texture(msaa: bool = false)
int	get_view_count() const
bool	has_texture(context: StringName, name: StringName) const

Method Descriptions

void **clear_context**(context: StringName) 🔗

Frees all buffers related to this context.

RID **create_texture**(context: StringName, name: StringName, data_format: DataFormat, usage_bits: int, texture_samples:

TextureSamples, size: Vector2i, layers: int, mipmaps: int, unique: bool) 🔗

Create a new texture with the given definition and cache this under the given name. Will return the existing texture if it already exists.

RID **create_texture_from_format**(context: StringName, name: StringName, format: RDTextureFormat, view: RDTextureView, unique: bool) 🔗

Create a new texture using the given format and view and cache this under the given name. Will return the existing texture if it already exists.

RID **create_texture_view**(context: StringName, name: StringName, view_name: StringName, view: RDTextureView) 🔗

Create a new texture view for an existing texture and cache this under the given view_name. Will return the existing teture view if it already exists. Will error if the source texture doesn't exist.

RID **get_color_layer**(layer: int, msaa: bool = false) 🔗

Returns the specified layer from the color texture we are rendering 3D content to.

If `msaa` is **true** and MSAA is enabled, this returns the MSAA variant of the buffer.

RID **get_color_texture**(msaa: bool = false) 🔗

Returns the color texture we are rendering 3D content to. If multiview is used this will be a texture array with all views.

If `msaa` is **true** and MSAA is enabled, this returns the MSAA variant of the buffer.

RID **get_depth_layer**(layer: int, msaa: bool = false) 🔗

Returns the specified layer from the depth texture we are rendering 3D content to.

If `msaa` is **true** and MSAA is enabled, this returns the MSAA variant of the buffer.

RID **get_depth_texture**(msaa: bool = false) 🔗

Returns the depth texture we are rendering 3D content to. If multiview is used this will be a texture array with all views.

If `msaa` is **true** and MSAA is enabled, this returns the MSAA variant of the buffer.

---

float **get_fsr_sharpness**() const 🔗

Returns the FSR sharpness value used while rendering the 3D content (if get_scaling_3d_mode is an FSR mode).

Vector2i **get_internal_size**() const 🔗

Returns the internal size of the render buffer (size before upscaling) with which textures are created by default.

ViewportMSAA **get_msaa_3d**() const 🔗

Returns the applied 3D MSAA mode for this viewport.

RID **get_render_target**() const 🔗

Returns the render target associated with this buffers object.

ViewportScaling3DMode **get_scaling_3d_mode**() const 🔗

Returns the scaling mode used for upscaling.

ViewportScreenSpaceAA **get_screen_space_aa**() const 🔗

Returns the screen-space antialiasing method applied.

Vector2i **get_target_size**() const 🔗

Returns the target size of the render buffer (size after upscaling).

RID **get_texture**(context: StringName, name: StringName) const 🔗

Returns a cached texture with this name.

RDTextureFormat **get_texture_format**(context: StringName, name: StringName) const 🔗

Returns the texture format information with which a cached texture was created.

TextureSamples **get_texture_samples**() const 🔗

Returns the number of MSAA samples used.

RID **get_texture_slice**(context: StringName, name: StringName, layer: int, mipmap: int, layers: int, mipmaps: int) 🔗

Returns a specific slice (layer or mipmap) for a cached texture.

Vector2i **get_texture_slice_size**(context: StringName, name: StringName, mipmap: int) 🔗

Returns the texture size of a given slice of a cached texture.

RID **get_texture_slice_view**(context: StringName, name: StringName, layer: int, mipmap: int, layers: int, mipmaps: int, view: RDTextureView) 🔗

Returns a specific view of a slice (layer or mipmap) for a cached texture.

bool **get_use_debanding**() const 🔗

Returns `true` if debanding is enabled.

bool **get_use_taa**() const 🔗

Returns `true` if TAA is enabled.

RID **get_velocity_layer**(layer: int, msaa: bool = false) 🔗

Returns the specified layer from the velocity texture we are rendering 3D content to.

RID **get_velocity_texture**(msaa: bool = false) 🔗

Returns the velocity texture we are rendering 3D content to. If multiview is used this will be a texture array with all views.

If `msaa` is **true** and MSAA is enabled, this returns the MSAA variant of the buffer.

int **get_view_count**() const 🔗

Returns the view count for the associated viewport.

bool **has_texture**(context: StringName, name: StringName) const 🔗

Returns `true` if a cached texture exists for this name.

RenderSceneData

Inherits: Object

Inherited By: RenderSceneDataExtension, RenderSceneDataRD

Abstract render data object, holds scene data related to rendering a single frame of a viewport.

Description

Abstract scene data object, exists for the duration of rendering a single viewport.

Note: This is an internal rendering server object, do not instantiate this from script.

Methods

Projection	get_cam_projection() const
Transform3D	get_cam_transform() const
RID	get_uniform_buffer() const
int	get_view_count() const
Vector3	get_view_eye_offset(view: int) const

Projection	get_view_projection(view: int) const

Method Descriptions

Projection **get_cam_projection**() const 🔗

Returns the camera projection used to render this frame.

Note: If more than one view is rendered, this will return a combined projection.

Transform3D **get_cam_transform**() const 🔗

Returns the camera transform used to render this frame.

Note: If more than one view is rendered, this will return a centered transform.

RID **get_uniform_buffer**() const 🔗

Return the RID of the uniform buffer containing the scene data as a UBO.

int **get_view_count**() const 🔗

Returns the number of views being rendered.

Vector3 **get_view_eye_offset**(view: int) const 🔗

Returns the eye offset per view used to render this frame. This is the offset between our camera transform and the eye transform.

Projection **get_view_projection**(view: int) const

Returns the view projection per view used to render this frame.

Note: If a single view is rendered, this returns the camera projection. If more than one view is rendered, this will return a projection for the given view including the eye offset.

RenderSceneDataExtension

Inherits: RenderSceneData < Object

This class allows for a RenderSceneData implementation to be made in GDExtension.

Description

This class allows for a RenderSceneData implementation to be made in GDExtension.

Methods

Projection	_get_cam_projection() virtual const
Transform3D	_get_cam_transform() virtual const
RID	_get_uniform_buffer() virtual const
int	_get_view_count() virtual const
Vector3	_get_view_eye_offset(view: int) virtual const
Projection	_get_view_projection(view: int) virtual const

Method Descriptions

Projection **_get_cam_projection**() virtual const 🔗

Implement this in GDExtension to return the camera Projection.

Transform3D **_get_cam_transform**() virtual const 🔗

Implement this in GDExtension to return the camera Transform3D.

RID **_get_uniform_buffer**() virtual const 🔗

Implement this in GDExtension to return the RID of the uniform buffer containing the scene data as a UBO.

---

int **_get_view_count**() virtual const 🔗

Implement this in GDExtension to return the view count.

Vector3 **_get_view_eye_offset**(view: int) virtual const 🔗

Implement this in GDExtension to return the eye offset for the given `view`.

Projection **_get_view_projection**(view: int) virtual const 🔗

Implement this in GDExtension to return the view Projection for the given `view`.

RenderSceneDataRD

Inherits: RenderSceneData **<** Object

Render scene data implementation for the RenderingDevice based renderers.

Description

Object holds scene data related to rendering a single frame of a viewport.

Note: This is an internal rendering server object, do not instantiate this from script.

ResourceFormatLoader

Inherits: RefCounted **<** Object

Loads a specific resource type from a file.

Description

Godot loads resources in the editor or in exported games using ResourceFormatLoaders. They are queried automatically via the ResourceLoader singleton, or when a resource with internal dependencies is loaded. Each file type may load as a different resource type, so multiple ResourceFormatLoaders are registered in the engine.

Extending this class allows you to define your own loader. Be sure to respect the documented return types and values. You should give it a global class name with `class_name` for it to be registered. Like built-in ResourceFormatLoaders, it will be called automatically when loading resources of its handled type(s). You may also implement a ResourceFormatSaver.

Note: You can also extend EditorImportPlugin if the resource type you need exists but Godot is unable to load its format. Choosing one way over another depends on if the format is suitable or not for the final exported game. For example, it's better to import `.png` textures as `.ctex` (CompressedTexture2D) first, so they can be loaded with better efficiency on the graphics card.

Methods

bool	_exists(path: String) virtual const
PackedStringArray	_get_classes_used(path: String) virtual const
PackedStringArray	_get_dependencies(path: String, add_types: bool) virtual const
PackedStringArray	_get_recognized_extensions() virtual const
String	_get_resource_script_class(path: String) virtual const
String	_get_resource_type(path: String) virtual const
int	_get_resource_uid(path: String) virtual const
bool	_handles_type(type: StringName) virtual const
Variant	_load(path: String, original_path: String, use_sub_threads: bool, cache_mode: int) virtual const
bool	_recognize_path(path: String, type: StringName) virtual const
Error	_rename_dependencies(path: String, renames: Dictionary) virtual const

---

Enumerations

enum **CacheMode**: 🔗

CacheMode **CACHE_MODE_IGNORE** = `0`

Neither the main resource (the one requested to be loaded) nor any of its subresources are retrieved from cache nor stored into it. Dependencies (external resources) are loaded with CACHE_MODE_REUSE.

CacheMode **CACHE_MODE_REUSE** = `1`

The main resource (the one requested to be loaded), its subresources, and its dependencies (external resources) are retrieved from cache if present, instead of loaded. Those not cached are loaded and then stored into the cache. The same rules are propagated recursively down the tree of dependencies (external resources).

CacheMode **CACHE_MODE_REPLACE** = `2`

Like CACHE_MODE_REUSE, but the cache is checked for the main resource (the one requested to be loaded) as well as for each of its subresources. Those already in the cache, as long as the loaded and cached types match, have their data refreshed from storage into the already existing instances. Otherwise, they are recreated as completely new objects.

CacheMode **CACHE_MODE_IGNORE_DEEP** = `3`

Like CACHE_MODE_IGNORE, but propagated recursively down the tree of dependencies (external resources).

CacheMode **CACHE_MODE_REPLACE_DEEP** = `4`

Like CACHE_MODE_REPLACE, but propagated recursively down the tree of dependencies (external resources).

Method Descriptions

bool **_exists**(path: String) virtual const

There is currently no description for this method. Please help us by contributing one!

PackedStringArray **_get_classes_used**(path: String) virtual const

There is currently no description for this method. Please help us by contributing one!

PackedStringArray **_get_dependencies**(path: String, add_types: bool) virtual const

If implemented, gets the dependencies of a given resource. If `add_types` is `true`, paths should be appended `::TypeName`, where `TypeName` is the class name of the dependency.

Note: Custom resource types defined by scripts aren't known by the ClassDB, so you might just return `"Resource"` for them.

PackedStringArray **_get_recognized_extensions**() virtual const

Gets the list of extensions for files this loader is able to read.

String **_get_resource_script_class**(path: String) virtual const 🔗

Returns the script class name associated with the Resource under the given `path`. If the resource has no script or the script isn't a named class, it should return `""`.

String **_get_resource_type**(path: String) virtual const 🔗

Gets the class name of the resource associated with the given path. If the loader cannot handle it, it should return `""`.

Note: Custom resource types defined by scripts aren't known by the ClassDB, so you might just return `"Resource"` for them.

---

int **_get_resource_uid**(path: String) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_handles_type**(type: StringName) virtual const 🔗

Tells which resource class this loader can load.

Note: Custom resource types defined by scripts aren't known by the ClassDB, so you might just handle `"Resource"` for them.

Variant **_load**(path: String, original_path: String, use_sub_threads: bool, cache_mode: int) virtual const 🔗

Loads a resource when the engine finds this loader to be compatible. If the loaded resource is the result of an import, `original_path` will target the source file. Returns a Resource object on success, or an Error constant in case of failure.

The `cache_mode` property defines whether and how the cache should be used or updated when loading the resource. See CacheMode for details.

---

bool **_recognize_path**(path: String, type: StringName) virtual const

Tells whether or not this loader should load a resource from its resource path for a given type.

If it is not implemented, the default behavior returns whether the path's extension is within the ones provided by _get_recognized_extensions, and if the type is within the ones provided by _get_resource_type.

Error **_rename_dependencies**(path: String, renames: Dictionary) virtual const

If implemented, renames dependencies within the given resource and saves it. `renames` is a dictionary `{ String => String }` mapping old dependency paths to new paths.

Returns @GlobalScope.OK on success, or an Error constant in case of failure.

ResourceFormatSaver

Inherits: RefCounted **<** Object

Saves a specific resource type to a file.

Description

The engine can save resources when you do it from the editor, or when you use the ResourceSaver singleton. This is accomplished thanks to multiple **ResourceFormatSaver**s, each handling its own format and called automatically by the engine.

By default, Godot saves resources as `.tres` (text-based), `.res` (binary) or another built-in format, but you can choose to create your own format by extending this class. Be sure to respect the documented return types and values. You should give it a global class name with `class_name` for it to be registered. Like built-in ResourceFormatSavers, it will be called automatically when saving resources of its recognized type(s). You may also implement a ResourceFormatLoader.

Methods

PackedStringArray	_get_recognized_extensions(resource: Resource) virtual const
bool	_recognize(resource: Resource) virtual const

bool	_recognize_path(resource: Resource, path: String) virtual const
Error	_save(resource: Resource, path: String, flags: int) virtual
Error	_set_uid(path: String, uid: int) virtual

Method Descriptions

PackedStringArray **_get_recognized_extensions**(resource: Resource) virtual const 🔗

Returns the list of extensions available for saving the resource object, provided it is recognized (see _recognize).

bool **_recognize**(resource: Resource) virtual const 🔗

Returns whether the given resource object can be saved by this saver.

bool **_recognize_path**(resource: Resource, path: String) virtual const 🔗

Returns `true` if this saver handles a given save path and `false` otherwise.

If this method is not implemented, the default behavior returns whether the path's extension is within the ones provided by _get_recognized_extensions.

Error **_save**(resource: Resource, path: String, flags: int) virtual 🔗

Saves the given resource object to a file at the target `path`. `flags` is a bitmask composed with SaverFlags constants.

Returns @GlobalScope.OK on success, or an Error constant in case of failure.

Error **_set_uid**(path: String, uid: int) virtual 🔗

Sets a new UID for the resource at the given `path`. Returns @GlobalScope.OK on success, or an Error constant in case of failure.

ResourceImporter

Inherits: RefCounted **<** Object

Inherited By: EditorImportPlugin, ResourceImporterBitMap, ResourceImporterBMFont, ResourceImporterCSVTranslation, ResourceImporterDynamicFont, ResourceImporterImage, ResourceImporterImageFont, ResourceImporterLayeredTexture, ResourceImporterMP3, ResourceImporterOBJ, ResourceImporterOggVorbis, ResourceImporterScene, ResourceImporterShaderFile, ResourceImporterTexture, ResourceImporterTextureAtlas, ResourceImporterWAV

Base class for resource importers.

Description

This is the base class for Godot's resource importers. To implement your own resource importers using editor plugins, see EditorImportPlugin.

Tutorials

- Import plugins

Enumerations

enum **ImportOrder**: 🔗

ImportOrder **IMPORT_ORDER_DEFAULT** = `0`

The default import order.

ImportOrder **IMPORT_ORDER_SCENE** = `100`

The import order for scenes, which ensures scenes are imported *after* all other core resources such as textures. Custom importers should generally have an import order lower than `100` to avoid issues when importing scenes that rely on custom resources.

ResourceImporterBitMap

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports a BitMap resource (2D array of boolean values).

Description

BitMap resources are typically used as click masks in TextureButton and TouchScreenButton.

Tutorials

- Importing images

Properties

int	create_from	`0`
float	threshold	`0.5`

Property Descriptions

int **create_from** = `0` 🔗

The data source to use for generating the bitmap.

Black & White: Pixels whose HSV value is greater than the threshold will be considered as "enabled" (bit is `true`). If the pixel is lower than or equal to the threshold, it will be considered as "disabled" (bit is `false`).

Alpha: Pixels whose alpha value is greater than the threshold will be considered as "enabled" (bit is `true`). If the pixel is lower than or equal to the threshold, it will be considered as "disabled" (bit is `false`).

---

float **threshold** = `0.5`

The threshold to use to determine which bits should be considered enabled or disabled. See also create_from.

ResourceImporterBMFont

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports a bitmap font in the BMFont (`.fnt`) format.

Description

The BMFont format is a format created by the BMFont [https://www.angelcode.com/products/bmfont/] program. Many BMFont-compatible programs also exist, like BMGlyph [https://www.bmglyph.com/].

Compared to ResourceImporterImageFont, **ResourceImporterBMFont** supports bitmap fonts with varying glyph widths/heights.

See also ResourceImporterDynamicFont.

Tutorials

- Bitmap fonts - Using fonts

Properties

bool	compress	`true`
Array	fallbacks	`[]`

int	scaling_mode	`2`

Property Descriptions

bool **compress** = `true`

If `true`, uses lossless compression for the resulting font.

Array **fallbacks** = `[]`

List of font fallbacks to use if a glyph isn't found in this bitmap font. Fonts at the beginning of the array are attempted first.

int **scaling_mode** = `2`

Font scaling mode.

ResourceImporterCSVTranslation

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports comma-separated values

Description

Comma-separated values are a plain text table storage format. The format's simplicity makes it easy to edit in any text editor or spreadsheet software. This makes it a common choice for game localization.

Example CSV file:

```
keys,en,es,ja
GREET,"Hello, friend!","Hola, amigo!",こんにちは
ASK,How are you?,Cómo está?,元気ですか
BYE,Goodbye,Adiós,さようなら
QUOTE,"""Hello"" said the man.","""Hola"" dijo el hombre.",「こんにちは」男は言いました
```

Tutorials

- Importing translations

Properties

bool	compress	`true`

int	delimiter	`0`

Property Descriptions

bool **compress** = `true`

If `true`, creates an OptimizedTranslation instead of a Translation. This makes the resulting file smaller at the cost of a small CPU overhead.

---

int **delimiter** = `0`

The delimiter to use in the CSV file. The default value matches the common CSV convention. Tab-separated values are sometimes called TSV files.

ResourceImporterDynamicFont

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports a TTF, TTC, OTF, OTC, WOFF or WOFF2 font file for font rendering that adapts to any size.

Description

Unlike bitmap fonts, dynamic fonts can be resized to any size and still look crisp. Dynamic fonts also optionally support MSDF font rendering, which allows for run-time scale changes with no re-rasterization cost.

While WOFF and especially WOFF2 tend to result in smaller file sizes, there is no universally "better" font format. In most situations, it's recommended to use the font format that was shipped on the font developer's website.

See also ResourceImporterBMFont and ResourceImporterImageFont.

Tutorials

- Dynamic fonts - Using fonts

Properties

bool	allow_system_fallback	`true`

int	antialiasing	`1`
bool	compress	`true`
bool	disable_embedded_bitmaps	`true`
Array	fallbacks	`[]`
bool	force_autohinter	`false`
bool	generate_mipmaps	`false`
int	hinting	`1`
Dictionary	language_support	`{}`
int	msdf_pixel_range	`8`
int	msdf_size	`48`
bool	multichannel_signed_distance_field	`false`
Dictionary	opentype_features	`{}`

float	oversampling	`0.0`
Array	preload	`[]`
Dictionary	script_support	`{}`
int	subpixel_positioning	`1`

Property Descriptions

bool **allow_system_fallback** = `true` 🔗

If `true`, automatically use system fonts as a fallback if a glyph isn't found in this dynamic font. This makes supporting CJK characters or emoji more straightforward, as you don't need to include a CJK/emoji font in your project. See also fallbacks.

Note: The appearance of system fonts varies across platforms. Loading system fonts is only supported on Windows, macOS, Linux, Android and iOS.

---

int **antialiasing** = `1` 🔗

The font antialiasing method to use.

Disabled: Most suited for pixel art fonts, although you do not *have* to change the antialiasing from the default **Grayscale** if the font file was well-created and the font is used at an integer multiple of its intended size. If pixel art fonts have a bad appearance at their intended size, try setting subpixel_positioning to **Disabled** instead.

Grayscale: Use grayscale antialiasing. This is the approach used by the operating system on macOS, Android and iOS.

LCD Subpixel: Use antialiasing with subpixel patterns to make fonts sharper on LCD displays. This is the approach used by the operating system on Windows and most Linux distributions. The downside is that this can introduce "fringing" on edges, especially on display technologies that don't use standard RGB subpixels (such as OLED displays). The LCD subpixel layout is globally controlled by ProjectSettings.gui/theme/lcd_subpixel_layout, which also allows falling back to grayscale antialiasing.

---

bool **compress** = `true`

If `true`, uses lossless compression for the resulting font.

---

bool **disable_embedded_bitmaps** = `true`

If set to `true`, embedded font bitmap loading is disabled (bitmap-only and color fonts ignore this property).

Array **fallbacks** = `[]`

List of font fallbacks to use if a glyph isn't found in this dynamic font. Fonts at the beginning of the array are attempted first, but fallback fonts that don't support the glyph's language and script are attempted last (see language_support and script_support). See also allow_system_fallback.

---

bool **force_autohinter** = `false` 🔗

If `true`, forces generation of hinting data for the font using FreeType [https://freetype.org/]'s autohinter. This will make hinting effective with fonts that don't include hinting data.

---

bool **generate_mipmaps** = `false` 🔗

If `true`, this font will have mipmaps generated. This prevents text from looking grainy when a Control is scaled down, or when a Label3D is viewed from a long distance (if Label3D.texture_filter is set to a mode that displays mipmaps).

Enabling generate_mipmaps increases font generation time and memory usage. Only enable this setting if you actually need it.

---

int **hinting** = `1` 🔗

The hinting mode to use. This controls how aggressively glyph edges should be snapped to pixels when rasterizing the font. Depending on personal preference, you may prefer using one hinting mode over the other. Hinting modes other than **None** are only effective if the font contains hinting data (see force_autohinter).

None: Smoothest appearance, which can make the font look blurry at small sizes.

Light: Sharp result by snapping glyph edges to pixels on the Y axis only.

Full: Sharpest by snapping glyph edges to pixels on both X and Y axes.

Dictionary **language_support** = `{}` 🔗

Override the list of languages supported by this font. If left empty, this is supplied by the font metadata. There is usually no need to change this. See also script_support.

int **msdf_pixel_range** = `8`

The width of the range around the shape between the minimum and maximum representable signed distance. If using font outlines, msdf_pixel_range must be set to at least *twice* the size of the largest font outline. The default msdf_pixel_range value of `8` allows outline sizes up to `4` to look correct.

int **msdf_size** = `48`

Source font size used to generate MSDF textures. Higher values allow for more precision, but are slower to render and require more memory. Only increase this value if you notice a visible lack of precision in glyph rendering. Only effective if multichannel_signed_distance_field is `true`.

bool **multichannel_signed_distance_field** = `false`

If set to `true`, the default font will use multichannel signed distance field (MSDF) for crisp rendering at any size. Since this approach does not rely on rasterizing the font every time its size changes, this allows for resizing the font in real-time without any performance penalty. Text will also not look grainy for Controls that are scaled down (or for Label3Ds viewed from a long distance).

MSDF font rendering can be combined with generate_mipmaps to further improve font rendering quality when scaled down.

Dictionary **opentype_features** = `{}`

The OpenType features to enable, disable or set a value for this font. This can be used to enable optional features provided by the font, such as ligatures or alternative glyphs. The list of supported OpenType features varies on a per-font basis.

float **oversampling** = `0.0`

If set to a value greater than `0.0`, overrides the oversampling factor for the font. This can be used to render the font at a higher or lower resolution than intended without affecting its physical size. In most cases, this should be left at `0.0`.

Array **preload** = `[]`

The glyph ranges to prerender. This can avoid stuttering during gameplay when new characters need to be rendered, especially if subpixel_positioning is enabled. The downside of using preloading is that initial project load times will increase, as well as memory usage.

Dictionary **script_support** = `{}`

Override the list of language scripts supported by this font. If left empty, this is supplied by the font metadata. There is usually no need to change this. See also language_support.

int **subpixel_positioning** = `1`

Subpixel positioning improves font rendering appearance, especially at smaller font sizes. The downside is that it takes more time to initially render the font, which can cause stuttering during gameplay, especially if used with large font sizes. This should be set to **Disabled** for fonts with a pixel art appearance.

Disabled: No subpixel positioning. Lowest quality, fastest rendering.

Auto: Use subpixel positioning at small font sizes (the chosen quality varies depending on font size). Large fonts will not use subpixel positioning. This is a good tradeoff between performance and quality.

One Half of a Pixel: Always perform intermediate subpixel positioning regardless of font size. High quality, slow rendering.

One Quarter of a Pixel: Always perform precise subpixel positioning regardless of font size. Highest quality, slowest rendering.

ResourceImporterImage

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports a image for use in scripting, with no rendering capabilities.

Description

This importer imports Image resources, as opposed to CompressedTexture2D. If you need to render the image in 2D or 3D, use ResourceImporterTexture instead.

Tutorials

- Importing images

ResourceImporterImageFont

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports a bitmap font where all glyphs have the same width and height.

Description

This image-based workflow can be easier to use than ResourceImporterBMFont, but it requires all glyphs to have the same width and height, glyph advances and drawing offsets can be customized. This makes **ResourceImporterImageFont** most suited to fixed-width fonts.

See also ResourceImporterDynamicFont.

Tutorials

- Bitmap fonts - Using fonts

Properties

int	ascent	`0`
Rect2i	character_margin	`Rect2i(0, 0, 0, 0)`
PackedStringArray	character_ranges	`PackedStringArray()`

int	columns	`1`
bool	compress	`true`
int	descent	`0`
Array	fallbacks	`[]`
Rect2i	image_margin	`Rect2i(0, 0, 0, 0)`
PackedStringArray	kerning_pairs	`PackedStringArray()`
int	rows	`1`
int	scaling_mode	`2`

Property Descriptions

int **ascent** = `0` 🔗

Font ascent (number of pixels above the baseline). If set to `0`, half of the character height is used.

Rect2i **character_margin** = `Rect2i(0, 0, 0, 0)` 🔗

Margin applied around every imported glyph. If your font image contains guides (in the form of lines between glyphs) or if spacing between characters appears incorrect, try adjusting character_margin.

PackedStringArray **character_ranges** = `PackedStringArray()` 🔗

The character ranges to import from the font image. This is an array that maps each position on the image (in tile coordinates, not pixels). The font atlas is traversed from left to right and top to bottom. Characters can be specified with decimal numbers (127), hexadecimal numbers (`0x007f`, or `U+007f`) or between single quotes (`'~'`). Ranges can be specified with a hyphen between characters.

For example, `0-127` represents the full ASCII range. It can also be written as `0x0000-0x007f` (or `U+0000-U+007f`). As another example, `' '-'~'` is equivalent to `32-127` and represents the range of printable (visible) ASCII characters.

For any range, the character advance and offset can be customized by appending three space-separated integer values (additional advance, x offset, y offset) to the end. For example `'a'-'b' 4 5 2` sets the advance to `char_width + 4` and offset to `Vector2(5, 2)` for both *a* and *b* characters.

Make sure character_ranges doesn't exceed the number of columns * rows defined. Otherwise, the font will fail to import.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedStringArray for more details.

---

int **columns** = `1` 🔗

Number of columns in the font image. See also rows.

bool **compress** = `true`

If `true`, uses lossless compression for the resulting font.

int **descent** = `0`

Font descent (number of pixels below the baseline). If set to `0`, half of the character height is used.

Array **fallbacks** = `[]`

List of font fallbacks to use if a glyph isn't found in this bitmap font. Fonts at the beginning of the array are attempted first.

Rect2i **image_margin** = `Rect2i(0, 0, 0, 0)`

Margin to cut on the sides of the entire image. This can be used to cut parts of the image that contain attribution information or similar.

PackedStringArray **kerning_pairs** = `PackedStringArray()`

Kerning pairs for the font. Kerning pair adjust the spacing between two characters.

Each string consist of three space separated values: "from" string, "to" string and integer offset. Each combination form the two string for a kerning pair, e.g, `ab cd -3` will create kerning pairs `ac`, `ad`, `bc`, and `bd` with offset `-3`. `\uXXXX` escape sequences can be used to add Unicode characters.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedStringArray for more details.

int **rows** = `1`

Number of rows in the font image. See also columns.

int **scaling_mode** = `2`

Font scaling mode.

ResourceImporterLayeredTexture

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports a 3-dimensional texture (Texture3D), a Texture2DArray, a Cubemap or a CubemapArray.

Description

This imports a 3-dimensional texture, which can then be used in custom shaders, as a FogMaterial density map or as a GPUParticlesAttractorVectorField3D. See also ResourceImporterTexture and ResourceImporterTextureAtlas.

Tutorials

- Importing images

Properties

int	compress/channel_pack	`0`
int	compress/hdr_compression	`1`
bool	compress/high_quality	`false`

float	compress/lossy_quality	`0.7`
int	compress/mode	`1`
bool	mipmaps/generate	`true`
int	mipmaps/limit	`-1`
int	slices/arrangement	`1`

Property Descriptions

int **compress/channel_pack** = `0` 🔗

Controls how color channels should be used in the imported texture.

sRGB Friendly:, prevents the RG color format from being used, as it does not support sRGB color.

Optimized:, allows the RG color format to be used if the texture does not use the blue channel. This reduces memory usage if the texture's blue channel can be discarded (all pixels must have a blue value of `0`).

Normal Map (RG Channels): This forces all layers from the texture to be imported with the RG color format to reduce memory usage, with only the red and green channels preserved. This only has an effect on textures with the VRAM Compressed or Basis

Universal compression modes. This mode is only available in layered textures (Cubemap, CubemapArray, Texture2DArray and Texture3D).

---

int **compress/hdr_compression** = `1` 🔗

Controls how VRAM compression should be performed for HDR images.

Disabled: Never use VRAM compression for HDR textures, regardless of whether they're opaque or transparent. Instead, the texture is converted to RGBE9995 (9-bits per channel + 5-bit exponent = 32 bits per pixel) to reduce memory usage compared to a half-float or single-precision float image format.

Opaque Only: Only uses VRAM compression for opaque HDR textures. This is due to a limitation of HDR formats, as there is no VRAM-compressed HDR format that supports transparency at the same time.

Always: Force VRAM compression even for HDR textures with an alpha channel. To perform this, the alpha channel is discarded on import.

Note: Only effective on Radiance HDR (`.hdr`) and OpenEXR (`.exr`) images.

---

bool **compress/high_quality** = `false` 🔗

If `true`, uses BPTC compression on desktop platforms and ASTC compression on mobile platforms. When using BPTC, BC7 is used for SDR textures and BC6H is used for HDR textures.

If `false`, uses the faster but lower-quality S3TC compression on desktop platforms and ETC2 on mobile/web platforms. When using

S3TC, DXT1 (BC1) is used for opaque textures and DXT5 (BC3) is used for transparent or normal map (RGTC) textures.

BPTC and ASTC support VRAM compression for HDR textures, but S3TC and ETC2 do not (see compress/hdr_compression).

---

float **compress/lossy_quality** = `0.7`

The quality to use when using the **Lossy** compression mode. Higher values result in better quality, at the cost of larger file sizes. Lossy quality does not affect memory usage of the imported texture, only its file size on disk.

---

int **compress/mode** = `1`

The compression mode to use. Each compression mode provides a different tradeoff:

Lossless: Original quality, high memory usage, high size on disk, fast import.

Lossy: Reduced quality, high memory usage, low size on disk, fast import.

VRAM Compressed: Reduced quality, low memory usage, low size on disk, slowest import. Only use for textures in 3D scenes, not for 2D elements.

VRAM Uncompressed: Original quality, high memory usage, highest size on disk, fastest import.

Basis Universal: Reduced quality, low memory usage, lowest size on disk, slow import. Only use for textures in 3D scenes, not for 2D elements.

See Compress mode in the manual for more details.

---

bool **mipmaps/generate** = `true` 🔗

If `true`, smaller versions of the texture are generated on import. For example, a 64×64 texture will generate 6 mipmaps (32×32, 16×16, 8×8, 4×4, 2×2, 1×1). This has several benefits:

- Textures will not become grainy in the distance (in 3D), or if scaled down due to Camera2D zoom or CanvasItem scale (in 2D).
- Performance will improve if the texture is displayed in the distance, since sampling smaller versions of the original texture is faster and requires less memory bandwidth.

The downside of mipmaps is that they increase memory usage by roughly 33% (for Texture2DArray, Cubemap and CubemapArray) or 14% (for Texture3D).

It's recommended to enable mipmaps in 3D. However, in 2D, this should only be enabled if your project visibly benefits from having mipmaps enabled. If the camera never zooms out significantly, there won't be a benefit to enabling mipmaps but memory usage will increase.

---

int **mipmaps/limit** = `-1` 🔗

Unimplemented. This currently has no effect when changed.

---

int **slices/arrangement** = `1` 🔗

Controls how the cubemap's texture is internally laid out. When using high-resolution cubemaps, **2×3** and **3×2** are less prone to exceeding hardware texture size limits compared to **1×6** and **6×1**.

ResourceImporterMP3

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports an MP3 audio file for playback.

Description

MP3 is a lossy audio format, with worse audio quality compared to ResourceImporterOggVorbis at a given bitrate.

In most cases, it's recommended to use Ogg Vorbis over MP3. However, if you're using an MP3 sound source with no higher quality source available, then it's recommended to use the MP3 file directly to avoid double lossy compression.

MP3 requires more CPU to decode than ResourceImporterWAV. If you need to play a lot of simultaneous sounds, it's recommended to use WAV for those sounds instead, especially if targeting low-end devices.

Tutorials

- Importing audio samples

Properties

int	bar_beats	`4`

int	beat_count	`0`
float	bpm	`0`
bool	loop	`false`
float	loop_offset	`0`

Property Descriptions

int **bar_beats** = `4` 🔗

The number of bars within a single beat in the audio track. This is only relevant for music that wishes to make use of interactive music functionality (not implemented yet), not sound effects.

A more convenient editor for bar_beats is provided in the **Advanced Import Settings** dialog, as it lets you preview your changes without having to reimport the audio.

---

int **beat_count** = `0` 🔗

The beat count of the audio track. This is only relevant for music that wishes to make use of interactive music functionality (not implemented yet), not sound effects.

A more convenient editor for beat_count is provided in the **Advanced Import Settings** dialog, as it lets you preview your changes without having to reimport the audio.

float **bpm** = `0`

The Beats Per Minute of the audio track. This should match the BPM measure that was used to compose the track. This is only relevant for music that wishes to make use of interactive music functionality (not implemented yet), not sound effects.

A more convenient editor for bpm is provided in the **Advanced Import Settings** dialog, as it lets you preview your changes without having to reimport the audio.

bool **loop** = `false`

If enabled, the audio will begin playing at the beginning after playback ends by reaching the end of the audio.

Note: In AudioStreamPlayer, the AudioStreamPlayer.finished signal won't be emitted for looping audio when it reaches the end of the audio file, as the audio will keep playing indefinitely.

float **loop_offset** = `0`

Determines where audio will start to loop after playback reaches the end of the audio. This can be used to only loop a part of the audio file, which is useful for some ambient sounds or music. The value is determined in seconds relative to the beginning of the audio. A value of `0.0` will loop the entire audio file.

Only has an effect if loop is `true`.

A more convenient editor for loop_offset is provided in the **Advanced Import Settings** dialog, as it lets you preview your changes without having to reimport the audio.

ResourceImporterOBJ

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports an OBJ 3D model as an independent Mesh or scene.

Description

Unlike ResourceImporterScene, **ResourceImporterOBJ** will import a single Mesh resource by default instead of importing a PackedScene. This makes it easier to use the Mesh resource in nodes that expect direct Mesh resources, such as GridMap, GPUParticles3D or CPUParticles3D. Note that it is still possible to save mesh resources from 3D scenes using the **Advanced Import Settings** dialog, regardless of the source format.

See also ResourceImporterScene, which is used for more advanced 3D formats such as glTF.

Tutorials

- Importing 3D scenes

Properties

bool	force_disable_mesh_compression	`false`
bool	generate_tangents	`true`

Vector3	offset_mesh	`Vector3(0, 0, 0)`
bool	optimize_mesh	`true`
Vector3	scale_mesh	`Vector3(1, 1, 1)`

Property Descriptions

bool **force_disable_mesh_compression** = `false` 🔗

If `true`, mesh compression will not be used. Consider enabling if you notice blocky artifacts in your mesh normals or UVs, or if you have meshes that are larger than a few thousand meters in each direction.

bool **generate_tangents** = `true` 🔗

If `true`, generate vertex tangents using Mikktspace [http://www.mikktspace.com/] if the source mesh doesn't have tangent data. When possible, it's recommended to let the 3D modeling software generate tangents on export instead on relying on this option. Tangents are required for correct display of normal and height maps, along with any material/shader features that require tangents.

If you don't need material features that require tangents, disabling this can reduce output file size and speed up importing if the source 3D file doesn't contain tangents.

Vector3 **offset_mesh** = `Vector3(0, 0, 0)`

Offsets the mesh's data by the specified value. This can be used to work around misaligned meshes without having to modify the source file.

---

bool **optimize_mesh** = `true`

Unused parameter. This currently has no effect.

Vector3 **scale_mesh** = `Vector3(1, 1, 1)`

Scales the mesh's data by the specified value. This can be used to work around misscaled meshes without having to modify the source file.

ResourceImporterOggVorbis

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports an Ogg Vorbis audio file for playback.

Description

Ogg Vorbis is a lossy audio format, with better audio quality compared to ResourceImporterMP3 at a given bitrate.

In most cases, it's recommended to use Ogg Vorbis over MP3. However, if you're using an MP3 sound source with no higher quality source available, then it's recommended to use the MP3 file directly to avoid double lossy compression.

Ogg Vorbis requires more CPU to decode than ResourceImporterWAV. If you need to play a lot of simultaneous sounds, it's recommended to use WAV for those sounds instead, especially if targeting low-end devices.

Tutorials

- Importing audio samples

Properties

int	bar_beats	`4`

int	beat_count	`0`
float	bpm	`0`
bool	loop	`false`
float	loop_offset	`0`

Methods

AudioStreamOggVorbis	load_from_buffer(buffer: PackedByteArray) static
AudioStreamOggVorbis	load_from_file(path: String) static

Property Descriptions

int **bar_beats** = `4` 🔗

The number of bars within a single beat in the audio track. This is only relevant for music that wishes to make use of interactive music functionality (not implemented yet), not sound effects.

A more convenient editor for bar_beats is provided in the **Advanced Import Settings** dialog, as it lets you preview your changes without having to reimport the audio.

int **beat_count** = `0`

The beat count of the audio track. This is only relevant for music that wishes to make use of interactive music functionality (not implemented yet), not sound effects.

A more convenient editor for beat_count is provided in the **Advanced Import Settings** dialog, as it lets you preview your changes without having to reimport the audio.

float **bpm** = `0`

The Beats Per Minute of the audio track. This should match the BPM measure that was used to compose the track. This is only relevant for music that wishes to make use of interactive music functionality (not implemented yet), not sound effects.

A more convenient editor for bpm is provided in the **Advanced Import Settings** dialog, as it lets you preview your changes without having to reimport the audio.

bool **loop** = `false`

If enabled, the audio will begin playing at the beginning after playback ends by reaching the end of the audio.

Note: In AudioStreamPlayer, the AudioStreamPlayer.finished signal won't be emitted for looping audio when it reaches the end of the audio file, as the audio will keep playing indefinitely.

float **loop_offset** = `0`

Determines where audio will start to loop after playback reaches the end of the audio. This can be used to only loop a part of the audio file, which is useful for some ambient sounds or music. The value is determined in seconds relative to the beginning of the audio. A value of `0.0` will loop the entire audio file.

Only has an effect if loop is `true`.

A more convenient editor for loop_offset is provided in the **Advanced Import Settings** dialog, as it lets you preview your changes without having to reimport the audio.

Method Descriptions

AudioStreamOggVorbis **load_from_buffer**(buffer: PackedByteArray) static 🔗

This method loads audio data from a PackedByteArray buffer into an AudioStreamOggVorbis object.

AudioStreamOggVorbis **load_from_file**(path: String) static 🔗

This method loads audio data from a file into an AudioStreamOggVorbis object. The file path is provided as a string.

ResourceImporterScene

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports a glTF, FBX, Collada or Blender 3D scene.

Description

See also ResourceImporterOBJ, which is used for OBJ models that can be imported as an independent Mesh or a scene.

Additional options (such as extracting individual meshes or materials to files) are available in the **Advanced Import Settings** dialog. This dialog can be accessed by double-clicking a 3D scene in the FileSystem dock or by selecting a 3D scene in the FileSystem dock, going to the Import dock and choosing **Advanced**.

Note: ResourceImporterScene is *not* used for PackedScenes, such as `.tscn` and `.scn` files.

Tutorials

- Importing 3D scenes

Properties

Dictionary	_subresources	`{}`
float	animation/fps	`30`

bool	animation/import	`true`
bool	animation/import_rest_as_RESET	`false`
bool	animation/remove_immutable_tracks	`true`
bool	animation/trimming	`false`
String	import_script/path	`""`
bool	meshes/create_shadow_meshes	`true`
bool	meshes/ensure_tangents	`true`
bool	meshes/force_disable_compression	`false`
bool	meshes/generate_lods	`true`
int	meshes/light_baking	`1`
float	meshes/lightmap_texel_size	`0.2`
bool	nodes/apply_root_scale	`true`

bool	nodes/import_as_skeleton_bones	`false`
String	nodes/root_name	`""`
float	nodes/root_scale	`1.0`
String	nodes/root_type	`""`
bool	skins/use_named_skins	`true`

Property Descriptions

Dictionary **_subresources** = `{}` 🔗

Contains properties for the scene's subresources. This is an internal option which is not visible in the Import dock.

---

float **animation/fps** = `30` 🔗

The number of frames per second to use for baking animation curves to a series of points with linear interpolation. It's recommended to configure this value to match the value you're using as a baseline in your 3D modeling software. Higher values result in more precise animation with fast movement changes, at the cost of higher file sizes and memory usage. Thanks to interpolation, there is usually not much benefit in going above 30 FPS (as the animation will still appear smooth at higher rendering framerates).

bool **animation/import** = `true`

If `true`, import animations from the 3D scene.

---

bool **animation/import_rest_as_RESET** = `false`

If `true`, adds an Animation named `RESET`, containing the Skeleton3D.get_bone_rest from Skeleton3D nodes. This can be useful to extract an animation in the reference pose.

---

bool **animation/remove_immutable_tracks** = `true`

If `true`, remove animation tracks that only contain default values. This can reduce output file size and memory usage with certain 3D scenes, depending on the contents of their animation tracks.

---

bool **animation/trimming** = `false`

If `true`, trim the beginning and end of animations if there are no keyframe changes. This can reduce output file size and memory usage with certain 3D scenes, depending on the contents of their animation tracks.

String **import_script/path** = `""`

Path to an import script, which can run code after the import process has completed for custom processing. See Using import scripts for automation for more information.

---

bool **meshes/create_shadow_meshes** = `true`

If `true`, enables the generation of shadow meshes on import. This optimizes shadow rendering without reducing quality by welding vertices together when possible. This in turn reduces the memory bandwidth required to render shadows. Shadow mesh generation currently doesn't support using a lower detail level than the source mesh (but shadow rendering will make use of LODs when relevant).

---

bool **meshes/ensure_tangents** = `true`

If `true`, generate vertex tangents using Mikktspace [http://www.mikktspace.com/] if the input meshes don't have tangent data. When possible, it's recommended to let the 3D modeling software generate tangents on export instead on relying on this option. Tangents are required for correct display of normal and height maps, along with any material/shader features that require tangents.

If you don't need material features that require tangents, disabling this can reduce output file size and speed up importing if the source 3D file doesn't contain tangents.

---

bool **meshes/force_disable_compression** = `false`

If `true`, mesh compression will not be used. Consider enabling if you notice blocky artifacts in your mesh normals or UVs, or if you have meshes that are larger than a few thousand meters in each direction.

---

bool **meshes/generate_lods** = `true`

If `true`, generates lower detail variants of the mesh which will be displayed in the distance to improve rendering performance. Not all meshes benefit from LOD, especially if they are never rendered from far away. Disabling this can reduce output file size and speed up importing. See Mesh level of detail (LOD) for more information.

int **meshes/light_baking** = `1` 🔗

Configures the meshes' GeometryInstance3D.gi_mode in the 3D scene. If set to **Static Lightmaps**, sets the meshes' GI mode to Static and generates UV2 on import for LightmapGI baking.

float **meshes/lightmap_texel_size** = `0.2` 🔗

Controls the size of each texel on the baked lightmap. A smaller value results in more precise lightmaps, at the cost of larger lightmap sizes and longer bake times.

Note: Only effective if meshes/light_baking is set to **Static Lightmaps**.

bool **nodes/apply_root_scale** = `true` 🔗

If `true`, nodes/root_scale will be applied to the descendant nodes, meshes, animations, bones, etc. This means that if you add a child node later on within the imported scene, it won't be scaled. If `false`, nodes/root_scale will multiply the scale of the root node instead.

bool **nodes/import_as_skeleton_bones** = `false` 🔗

Treat all nodes in the imported scene as if they are bones within a single Skeleton3D. Can be used to guarantee that imported animations target skeleton bones rather than nodes. May also be used to assign the `"Root"` bone in a BoneMap. See Retargeting 3D Skeletons for more information.

String **nodes/root_name** = `""` 🔗

Override for the root node name. If empty, the root node will use what the scene specifies, or the file name if the scene does not specify a root name.

---

float **nodes/root_scale** = `1.0` 🔗

The uniform scale to use for the scene root. The default value of `1.0` will not perform any rescaling. See nodes/apply_root_scale for details of how this scale is applied.

String **nodes/root_type** = `""` 🔗

Override for the root node type. If empty, the root node will use what the scene specifies, or Node3D if the scene does not specify a root type. Using a node type that inherits from Node3D is recommended. Otherwise, you'll lose the ability to position the node directly in the 3D editor.

---

bool **skins/use_named_skins** = `true` 🔗

If checked, use named Skins for animation. The MeshInstance3D node contains 3 properties of relevance here: a skeleton NodePath pointing to the Skeleton3D node (usually `..`), a mesh, and a skin:

- The Skeleton3D node contains a list of bones with names, their pose and rest, a name and a parent bone.
- The mesh is all of the raw vertex data needed to display a mesh. In terms of the mesh, it knows how vertices are weight-painted and uses some internal numbering often imported from 3D modeling software.

- The skin contains the information necessary to bind this mesh onto this Skeleton3D. For every one of the internal bone IDs chosen by the 3D modeling software, it contains two things. Firstly, a matrix known as the Bind Pose Matrix, Inverse Bind Matrix, or IBM for short. Secondly, the Skin contains each bone's name (if skins/use_named_skins is `true`), or the bone's index within the Skeleton3D list (if skins/use_named_skins is `false`).

Together, this information is enough to tell Godot how to use the bone poses in the Skeleton3D node to render the mesh from each MeshInstance3D. Note that each MeshInstance3D may share binds, as is common in models exported from Blender, or each MeshInstance3D may use a separate Skin object, as is common in models exported from other tools such as Maya.

ResourceImporterShaderFile

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports native GLSL shaders (not Godot shaders) as a RDShaderFile.

Description

This imports native GLSL shaders as RDShaderFile resources, for use with low-level RenderingDevice operations. This importer does *not* handle `.gdshader` files.

ResourceImporterTexture

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports an image for use in 2D or 3D rendering.

Description

This importer imports CompressedTexture2D resources. If you need to process the image in scripts in a more convenient way, use ResourceImporterImage instead. See also ResourceImporterLayeredTexture.

Tutorials

- Importing images

Properties

int	compress/channel_pack	`0`
int	compress/hdr_compression	`1`
bool	compress/high_quality	`false`
float	compress/lossy_quality	`0.7`

int	compress/mode	`0`
int	compress/normal_map	`0`
int	detect_3d/compress_to	`1`
bool	editor/convert_colors_with_editor_theme	`false`
bool	editor/scale_with_editor_scale	`false`
bool	mipmaps/generate	`false`
int	mipmaps/limit	`-1`
bool	process/fix_alpha_border	`true`
bool	process/hdr_as_srgb	`false`
bool	process/hdr_clamp_exposure	`false`
bool	process/normal_map_invert_y	`false`
bool	process/premult_alpha	`false`
int	process/size_limit	`0`

int	roughness/mode	`0`
String	roughness/src_normal	`""`
float	svg/scale	`1.0`

Property Descriptions

int **compress/channel_pack** = `0` 🔗

Controls how color channels should be used in the imported texture.

sRGB Friendly: Prevents the RG color format from being used, as it does not support sRGB color.

Optimized: Allows the RG color format to be used if the texture does not use the blue channel. This reduces memory usage if the texture's blue channel can be discarded (all pixels must have a blue value of `0`).

---

int **compress/hdr_compression** = `1` 🔗

Controls how VRAM compression should be performed for HDR images.

Disabled: Never use VRAM compression for HDR textures, regardless of whether they're opaque or transparent. Instead, the texture is converted to RGBE9995 (9-bits per channel + 5-bit exponent = 32 bits per pixel) to reduce memory usage compared to a half-float or single-precision float image format.

Opaque Only: Only uses VRAM compression for opaque HDR textures. This is due to a limitation of HDR formats, as there is no VRAM-compressed HDR format that supports transparency at the same time.

Always: Force VRAM compression even for HDR textures with an alpha channel. To perform this, the alpha channel is discarded on import.

Note: Only effective on Radiance HDR (`.hdr`) and OpenEXR (`.exr`) images.

---

bool **compress/high_quality** = `false`

If `true`, uses BPTC compression on desktop platforms and ASTC compression on mobile platforms. When using BPTC, BC7 is used for SDR textures and BC6H is used for HDR textures.

If `false`, uses the faster but lower-quality S3TC compression on desktop platforms and ETC2 on mobile/web platforms. When using S3TC, DXT1 (BC1) is used for opaque textures and DXT5 (BC3) is used for transparent or normal map (RGTC) textures.

BPTC and ASTC support VRAM compression for HDR textures, but S3TC and ETC2 do not (see compress/hdr_compression).

---

float **compress/lossy_quality** = `0.7`

The quality to use when using the **Lossy** compression mode. Higher values result in better quality, at the cost of larger file sizes. Lossy quality does not affect memory usage of the imported texture, only its file size on disk.

---

int **compress/mode** = 0

The compression mode to use. Each compression mode provides a different tradeoff:

Lossless: Original quality, high memory usage, high size on disk, fast import.

Lossy: Reduced quality, high memory usage, low size on disk, fast import.

VRAM Compressed: Reduced quality, low memory usage, low size on disk, slowest import. Only use for textures in 3D scenes, not for 2D elements.

VRAM Uncompressed: Original quality, high memory usage, highest size on disk, fastest import.

Basis Universal: Reduced quality, low memory usage, lowest size on disk, slow import. Only use for textures in 3D scenes, not for 2D elements.

See Compress mode in the manual for more details.

---

int **compress/normal_map** = 0

When using a texture as normal map, only the red and green channels are required. Given regular texture compression algorithms produce artifacts that don't look that nice in normal maps, the RGTC compression format is the best fit for this data. Forcing this option to Enable will make Godot import the image as RGTC compressed. By default, it's set to Detect. This means that if the texture is ever detected to be used as a normal map, it will be changed to Enable and reimported automatically.

Note that RGTC compression affects the resulting normal map image. You will have to adjust custom shaders that use the normal map's blue channel to take this into account. Built-in material shaders already ignore the blue channel in a normal map (regardless of the actual normal map's contents).

---

int **detect_3d/compress_to** = `1`

This changes the compress/mode option that is used when a texture is detected as being used in 3D.

Changing this import option only has an effect if a texture is detected as being used in 3D. Changing this to **Disabled** then reimporting will not change the existing compress mode on a texture (if it's detected to be used in 3D), but choosing **VRAM Compressed** or **Basis Universal** will.

---

bool **editor/convert_colors_with_editor_theme** = `false`

If `true`, converts the imported image's colors to match EditorSettings.interface/theme/icon_and_font_color. This assumes the image uses the exact same colors as Godot's own color palette for editor icons, with the source file designed for a dark editor theme. This should be enabled for editor plugin icons and custom class icons, but should be left disabled otherwise.

Note: Only available for SVG images.

---

bool **editor/scale_with_editor_scale** = `false`

If `true`, scales the imported image to match EditorSettings.interface/editor/custom_display_scale. This should be enabled for editor plugin icons and custom class icons, but should be left disabled otherwise.

Note: Only available for SVG images.

bool **mipmaps/generate** = `false`

If `true`, smaller versions of the texture are generated on import. For example, a 64×64 texture will generate 6 mipmaps (32×32, 16×16, 8×8, 4×4, 2×2, 1×1). This has several benefits:

- Textures will not become grainy in the distance (in 3D), or if scaled down due to Camera2D zoom or CanvasItem scale (in 2D).
- Performance will improve if the texture is displayed in the distance, since sampling smaller versions of the original texture is faster and requires less memory bandwidth.

The downside of mipmaps is that they increase memory usage by roughly 33%.

It's recommended to enable mipmaps in 3D. However, in 2D, this should only be enabled if your project visibly benefits from having mipmaps enabled. If the camera never zooms out significantly, there won't be a benefit to enabling mipmaps but memory usage will increase.

int **mipmaps/limit** = `-1`

Unimplemented. This currently has no effect when changed.

bool **process/fix_alpha_border** = `true`

If `true`, puts pixels of the same surrounding color in transition from transparent to opaque areas. For textures displayed with bilinear filtering, this helps to reduce the outline effect when exporting images from an image editor.

It's recommended to leave this enabled (as it is by default), unless this causes issues for a particular image.

bool **process/hdr_as_srgb** = `false` 🔗

Some HDR images you can find online may be broken and contain sRGB color data (instead of linear color data). It is advised not to use those files. If you absolutely have to, enabling process/hdr_as_srgb will make them look correct.

Warning: Enabling process/hdr_as_srgb on well-formatted HDR images will cause the resulting image to look too dark, so leave this on `false` if unsure.

bool **process/hdr_clamp_exposure** = `false` 🔗

If `true`, clamps exposure in the imported high dynamic range images using a smart clamping formula (without introducing *visible* clipping).

Some HDR panorama images you can find online may contain extremely bright pixels, due to being taken from real life sources without any clipping.

While these HDR panorama images are accurate to real life, this can cause the radiance map generated by Godot to contain sparkles when used as a background sky. This can be seen in material reflections (even on rough materials in extreme cases). Enabling process/hdr_clamp_exposure can resolve this.

---

bool **process/normal_map_invert_y** = `false` 🔗

If `true`, convert the normal map from Y- (DirectX-style) to Y+ (OpenGL-style) by inverting its green color channel. This is the normal map convention expected by Godot.

More information about normal maps (including a coordinate order table for popular engines) can be found here [http://wiki.polycount.com/wiki/Normal_Map_Technical_Details].

---

bool **process/premult_alpha** = `false` 🔗

An alternative to fixing darkened borders with process/fix_alpha_border is to use premultiplied alpha. By enabling this option, the texture will be converted to this format. A premultiplied alpha texture requires specific materials to be displayed correctly:

- In 2D, a CanvasItemMaterial will need to be created and configured to use the CanvasItemMaterial.BLEND_MODE_PREMULT_ALPHA blend mode on CanvasItems that use this texture. In custom `@canvas_item` shaders, `render_mode blend_premul_alpha;` should be used.
- In 3D, a BaseMaterial3D will need to be created and configured to use the BaseMaterial3D.BLEND_MODE_PREMULT_ALPHA blend mode on materials that use this texture. In custom `spatial` shaders, `render_mode blend_premul_alpha;` should be used.

---

int **process/size_limit** = `0` 🔗

If set to a value greater than `0`, the size of the texture is limited on import to a value smaller than or equal to the value specified here. For non-square textures, the size limit affects the longer dimension, with the shorter dimension scaled to preserve aspect ratio. Resizing is performed using cubic interpolation.

This can be used to reduce memory usage without affecting the source images, or avoid issues with textures not displaying on mobile/web platforms (as these usually can't display textures larger than 4096×4096).

int **roughness/mode** = `0`

The color channel to consider as a roughness map in this texture. Only effective if Roughness > Src Normal is not empty.

String **roughness/src_normal** = `""`

The path to the texture to consider as a normal map for roughness filtering on import. Specifying this can help decrease specular aliasing slightly in 3D.

Roughness filtering on import is only used in 3D rendering, not 2D.

float **svg/scale** = `1.0`

The scale the SVG should be rendered at, with `1.0` being the original design size. Higher values result in a larger image. Note that unlike font oversampling, this affects the size the SVG is rendered at in 2D. See also editor/scale_with_editor_scale.

Note: Only available for SVG images.

ResourceImporterTextureAtlas

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports a collection of textures from a PNG image into an optimized AtlasTexture for 2D rendering.

Description

This imports a collection of textures from a PNG image into an AtlasTexture or 2D ArrayMesh. This can be used to save memory when importing 2D animations from spritesheets. Texture atlases are only supported in 2D rendering, not 3D. See also ResourceImporterTexture and ResourceImporterLayeredTexture.

Note: ResourceImporterTextureAtlas does not handle importing TileSetAtlasSource, which is created using the TileSet editor instead.

Properties

String	atlas_file	`""`
bool	crop_to_region	`false`
int	import_mode	`0`

bool	trim_alpha_border_from_region	`true`

Property Descriptions

String **atlas_file** = `""`

Path to the atlas spritesheet. This *must* be set to valid path to a PNG image. Otherwise, the atlas will fail to import.

bool **crop_to_region** = `false`

If `true`, discards empty areas from the atlas. This only affects final sprite positioning, not storage. See also trim_alpha_border_from_region.

Note: Only effective if import_mode is **Region**.

int **import_mode** = `0`

Region: Imports the atlas in an AtlasTexture resource, which is rendered as a rectangle. This is fast to render, but transparent areas still have to be rendered if they can't be trimmed effectively by trim_alpha_border_from_region. This can reduce performance when rendering large sprites on screen.

Mesh: Imports the atlas as an ArrayMesh resource, keeping the original bitmap visible (but rendered as a polygon). This can be used to reduce fill rate when rendering large transparent sprites, at the cost of slower rendering if there are little to no transparent areas in the sprite.

bool **trim_alpha_border_from_region** = `true` 🔗

If `true`, trims the region to exclude fully transparent pixels using a clipping rectangle (which is never rotated). This can be used to save memory. See also crop_to_region.

Note: Only effective if import_mode is **Region**.

ResourceImporterWAV

Inherits: ResourceImporter **<** RefCounted **<** Object

Imports a WAV audio file for playback.

Description

WAV is an uncompressed format, which can provide higher quality compared to Ogg Vorbis and MP3. It also has the lowest CPU cost to decode. This means high numbers of WAV sounds can be played at the same time, even on low-end deviceS.

Tutorials

- Importing audio samples

Properties

int	compress/mode	`0`
int	edit/loop_begin	`0`
int	edit/loop_end	`-1`
int	edit/loop_mode	`0`

bool	edit/normalize	`false`
bool	edit/trim	`false`
bool	force/8_bit	`false`
bool	force/max_rate	`false`
float	force/max_rate_hz	`44100`
bool	force/mono	`false`

Property Descriptions

int **compress/mode** = `0` 🔗

The compression mode to use on import.

Disabled: Imports audio data without any compression. This results in the highest possible quality.

RAM (Ima-ADPCM): Performs fast lossy compression on import. Low CPU cost, but quality is noticeably decreased compared to Ogg Vorbis or even MP3.

QOA (`Quite OK Audio <https://qoaformat.org/>`__): Performs lossy compression on import. CPU cost is slightly higher than IMA-ADPCM, but quality is much higher.

---

int **edit/loop_begin** = `0` 🔗

The begin loop point to use when edit/loop_mode is **Forward**, **Ping-Pong**, or **Backward**. This is set in samples after the beginning of the audio file.

---

int **edit/loop_end** = `-1` 🔗

The end loop point to use when edit/loop_mode is **Forward**, **Ping-Pong**, or **Backward**. This is set in samples after the beginning of the audio file. A value of `-1` uses the end of the audio file as the end loop point.

---

int **edit/loop_mode** = `0` 🔗

Controls how audio should loop. This is automatically read from the WAV metadata on import.

Disabled: Don't loop audio, even if metadata indicates the file should be played back looping.

Forward: Standard audio looping.

Ping-Pong: Play audio forward until it's done playing, then play it backward and repeat. This is similar to mirrored texture repeat, but for audio.

Backward: Play audio in reverse and loop back to the end when done playing.

Note: In AudioStreamPlayer, the AudioStreamPlayer.finished signal won't be emitted for looping audio when it reaches the end of the audio file, as the audio will keep playing indefinitely.

---

bool **edit/normalize** = `false`

If `true`, normalize the audio volume so that its peak volume is equal to 0 dB. When enabled, normalization will make audio sound louder depending on its original peak volume.

bool **edit/trim** = `false`

If `true`, automatically trim the beginning and end of the audio if it's lower than -50 dB after normalization (see edit/normalize). This prevents having files with silence at the beginning or end, which increases their size unnecessarily and adds latency to the moment they are played back. A fade-in/fade-out period of 500 samples is also used during trimming to avoid audible pops.

bool **force/8_bit** = `false`

If `true`, forces the imported audio to use 8-bit quantization if the source file is 16-bit or higher.

Enabling this is generally not recommended, as 8-bit quantization decreases audio quality significantly. If you need smaller file sizes, consider using Ogg Vorbis or MP3 audio instead.

bool **force/max_rate** = `false`

If set to a value greater than `0`, forces the audio's sample rate to be reduced to a value lower than or equal to the value specified in force/max_rate_hz.

This can decrease file size noticeably on certain sounds, without impacting quality depending on the actual sound's contents. See Best practices for more information.

ResourceLoader

Inherits: Object

A singleton for loading resource files.

Description

A singleton used to load resource files from the filesystem.

It uses the many ResourceFormatLoader classes registered in the engine (either built-in or from a plugin) to load files into memory and convert them to a format that can be used by the engine.

Note: You have to import the files into the engine first to load them using load. If you want to load Images at run-time, you may use Image.load. If you want to import audio files, you can use the snippet described in AudioStreamMP3.data.

Tutorials

- Operating System Testing Demo [https://godotengine.org/asset-library/asset/2789]

Methods

void	add_resource_format_loader(format_loader: ResourceFormatLoader, at_front: bool = false)

bool	exists(path: String, type_hint: String = "")
PackedStringArray	get_dependencies(path: String)
PackedStringArray	get_recognized_extensions_for_type(type: String)
int	get_resource_uid(path: String)
bool	has_cached(path: String)
Resource	load(path: String, type_hint: String = "", cache_mode: CacheMode = 1)
Resource	load_threaded_get(path: String)
ThreadLoadStatus	load_threaded_get_status(path: String, progress: Array = [])
Error	load_threaded_request(path: String, type_hint: String = "", use_sub_threads: bool = false, cache_mode: CacheMode = 1)
void	remove_resource_format_loader(format_loader: ResourceFormatLoader)
void	set_abort_on_missing_resources(abort: bool)

Enumerations

enum **ThreadLoadStatus**: 🔗

ThreadLoadStatus **THREAD_LOAD_INVALID_RESOURCE** = `0`

The resource is invalid, or has not been loaded with load_threaded_request.

ThreadLoadStatus **THREAD_LOAD_IN_PROGRESS** = `1`

The resource is still being loaded.

ThreadLoadStatus **THREAD_LOAD_FAILED** = `2`

Some error occurred during loading and it failed.

ThreadLoadStatus **THREAD_LOAD_LOADED** = `3`

The resource was loaded successfully and can be accessed via load_threaded_get.

---

enum **CacheMode**: 🔗

CacheMode **CACHE_MODE_IGNORE** = `0`

Neither the main resource (the one requested to be loaded) nor any of its subresources are retrieved from cache nor stored into it. Dependencies (external resources) are loaded with CACHE_MODE_REUSE.

CacheMode **CACHE_MODE_REUSE** = `1`

The main resource (the one requested to be loaded), its subresources, and its dependencies (external resources) are retrieved from cache if present, instead of loaded. Those not cached are loaded and then stored into the cache. The same rules are propagated recursively down the tree of dependencies (external resources).

CacheMode **CACHE_MODE_REPLACE** = `2`

Like CACHE_MODE_REUSE, but the cache is checked for the main resource (the one requested to be loaded) as well as for each of its subresources. Those already in the cache, as long as the loaded and cached types match, have their data refreshed from storage into the already existing instances. Otherwise, they are recreated as completely new objects.

CacheMode **CACHE_MODE_IGNORE_DEEP** = `3`

Like CACHE_MODE_IGNORE, but propagated recursively down the tree of dependencies (external resources).

CacheMode **CACHE_MODE_REPLACE_DEEP** = `4`

Like CACHE_MODE_REPLACE, but propagated recursively down the tree of dependencies (external resources).

Method Descriptions

void **add_resource_format_loader**(format_loader: ResourceFormatLoader, at_front: bool = false) 🔗

Registers a new ResourceFormatLoader. The ResourceLoader will use the ResourceFormatLoader as described in load.

This method is performed implicitly for ResourceFormatLoaders written in GDScript (see ResourceFormatLoader for more information).

---

bool **exists**(path: String, type_hint: String = "") 🔗

Returns whether a recognized resource exists for the given `path`.

An optional `type_hint` can be used to further specify the Resource type that should be handled by the ResourceFormatLoader. Anything that inherits from Resource can be used as a type hint, for example Image.

Note: If you use Resource.take_over_path, this method will return `true` for the taken path even if the resource wasn't saved (i.e. exists only in resource cache).

PackedStringArray **get_dependencies**(path: String) 🔗

Returns the dependencies for the resource at the given `path`.

Note: The dependencies are returned with slices separated by `::`. You can use String.get_slice to get their components.

```
for dep in ResourceLoader.get_dependencies(path):
    print(dep.get_slice("::", 0)) # Prints UID.
    print(dep.get_slice("::", 2)) # Prints path.
```

PackedStringArray **get_recognized_extensions_for_type**(type: String) 🔗

Returns the list of recognized extensions for a resource type.

---

int **get_resource_uid**(path: String)

Returns the ID associated with a given resource path, or `-1` when no such ID exists.

---

bool **has_cached**(path: String)

Returns whether a cached resource is available for the given `path`.

Once a resource has been loaded by the engine, it is cached in memory for faster access, and future calls to the load method will use the cached version. The cached resource can be overridden by using Resource.take_over_path on a new resource for that same path.

Resource **load**(path: String, type_hint: String = "", cache_mode: CacheMode = 1)

Loads a resource at the given `path`, caching the result for further access.

The registered ResourceFormatLoaders are queried sequentially to find the first one which can handle the file's extension, and then attempt loading. If loading fails, the remaining ResourceFormatLoaders are also attempted.

An optional `type_hint` can be used to further specify the Resource type that should be handled by the ResourceFormatLoader. Anything that inherits from Resource can be used as a type hint, for example Image.

The `cache_mode` property defines whether and how the cache should be used or updated when loading the resource. See CacheMode for details.

Returns an empty resource if no ResourceFormatLoader could handle the file, and prints an error if no file is found at the specified path.

GDScript has a simplified @GDScript.load built-in method which can be used in most situations, leaving the use of **ResourceLoader** for more advanced scenarios.

Note: If ProjectSettings.editor/export/convert_text_resources_to_binary is `true`, @GDScript.load will not be able to read converted files in an exported project. If you rely on run-time loading of files present within the PCK, set ProjectSettings.editor/export/convert_text_resources_to_binary to `false`.

Note: Relative paths will be prefixed with `"res://"` before loading, to avoid unexpected results make sure your paths are absolute.

Resource **load_threaded_get**(path: String) 🔗

Returns the resource loaded by load_threaded_request.

If this is called before the loading thread is done (i.e. load_threaded_get_status is not THREAD_LOAD_LOADED), the calling thread will be blocked until the resource has finished loading. However, it's recommended to use load_threaded_get_status to known when the load has actually completed.

ThreadLoadStatus **load_threaded_get_status**(path: String, progress: Array = []) 🔗

Returns the status of a threaded loading operation started with load_threaded_request for the resource at `path`. See

ThreadLoadStatus for possible return values.

An array variable can optionally be passed via `progress`, and will return a one-element array containing the percentage of completion of the threaded loading.

Note: The recommended way of using this method is to call it during different frames (e.g., in Node._process, instead of a loop).

Error **load_threaded_request**(path: String, type_hint: String = "", use_sub_threads: bool = false, cache_mode: CacheMode = 1) 🔗

Loads the resource using threads. If `use_sub_threads` is `true`, multiple threads will be used to load the resource, which makes loading faster, but may affect the main thread (and thus cause game slowdowns).

The `cache_mode` property defines whether and how the cache should be used or updated when loading the resource. See CacheMode for details.

---

void **remove_resource_format_loader**(format_loader: ResourceFormatLoader) 🔗

Unregisters the given ResourceFormatLoader.

---

void **set_abort_on_missing_resources**(abort: bool) 🔗

Changes the behavior on missing sub-resources. The default behavior is to abort loading.

ResourceSaver

Inherits: Object

A singleton for saving Resources to the filesystem.

Description

A singleton for saving resource types to the filesystem.

It uses the many ResourceFormatSaver classes registered in the engine (either built-in or from a plugin) to save resource data to text-based (e.g. `.tres` or `.tscn`) or binary files (e.g. `.res` or `.scn`).

Methods

void	add_resource_format_saver(format_saver: ResourceFormatSaver, at_front: bool = false)
PackedStringArray	get_recognized_extensions(type: Resource)
void	remove_resource_format_saver(format_saver: ResourceFormatSaver)
Error	save(resource: Resource, path: String = "", flags: BitField[SaverFlags] = 0)

Enumerations

flags **SaverFlags**: 🔗

SaverFlags **FLAG_NONE** = `0`

No resource saving option.

SaverFlags **FLAG_RELATIVE_PATHS** = `1`

Save the resource with a path relative to the scene which uses it.

SaverFlags **FLAG_BUNDLE_RESOURCES** = `2`

Bundles external resources.

SaverFlags **FLAG_CHANGE_PATH** = `4`

Changes the Resource.resource_path of the saved resource to match its new location.

SaverFlags **FLAG_OMIT_EDITOR_PROPERTIES** = `8`

Do not save editor-specific metadata (identified by their `__editor` prefix).

SaverFlags **FLAG_SAVE_BIG_ENDIAN** = `16`

Save as big endian (see FileAccess.big_endian).

SaverFlags **FLAG_COMPRESS** = `32`

Compress the resource on save using FileAccess.COMPRESSION_ZSTD. Only available for binary resource types.

SaverFlags **FLAG_REPLACE_SUBRESOURCE_PATHS** = `64`

Take over the paths of the saved subresources (see Resource.take_over_path).

Method Descriptions

void **add_resource_format_saver**(format_saver: ResourceFormatSaver, at_front: bool = false)

Registers a new ResourceFormatSaver. The ResourceSaver will use the ResourceFormatSaver as described in save.

This method is performed implicitly for ResourceFormatSavers written in GDScript (see ResourceFormatSaver for more information).

PackedStringArray **get_recognized_extensions**(type: Resource)

Returns the list of extensions available for saving a resource of a given type.

void **remove_resource_format_saver**(format_saver: ResourceFormatSaver)

Unregisters the given ResourceFormatSaver.

Error **save**(resource: Resource, path: String = "", flags: BitField[SaverFlags] = 0)

Saves a resource to disk to the given path, using a ResourceFormatSaver that recognizes the resource object. If `path` is empty, **ResourceSaver** will try to use Resource.resource_path.

The `flags` bitmask can be specified to customize the save behavior using SaverFlags flags.

Returns @GlobalScope.OK on success.

Note: When the project is running, any generated UID associated with the resource will not be saved as the required code is only executed in editor mode.

ResourceUID

Inherits: Object

A singleton that manages the unique identifiers of all resources within a project.

Description

Resource UIDs (Unique IDentifiers) allow the engine to keep references between resources intact, even if files can renamed or moved. They can be accessed with `uid://`.

ResourceUID keeps track of all registered resource UIDs in a project, generates new UIDs, and converts between their string and integer representations.

Methods

void	add_id(id: int, path: String)
int	create_id()
String	get_id_path(id: int) const
bool	has_id(id: int) const
String	id_to_text(id: int) const

void	remove_id(id: int)
void	set_id(id: int, path: String)
int	text_to_id(text_id: String) const

Constants

INVALID_ID = `-1`

The value to use for an invalid UID, for example if the resource could not be loaded.

Its text representation is `uid://<invalid>`.

Method Descriptions

void **add_id**(id: int, path: String)

Adds a new UID value which is mapped to the given resource path.

Fails with an error if the UID already exists, so be sure to check has_id beforehand, or use set_id instead.

int **create_id**()

Generates a random resource UID which is guaranteed to be unique within the list of currently loaded UIDs.

In order for this UID to be registered, you must call add_id or set_id.

String **get_id_path**(id: int) const 🔗

Returns the path that the given UID value refers to.

Fails with an error if the UID does not exist, so be sure to check has_id beforehand.

---

bool **has_id**(id: int) const 🔗

Returns whether the given UID value is known to the cache.

String **id_to_text**(id: int) const 🔗

Converts the given UID to a `uid://` string value.

---

void **remove_id**(id: int) 🔗

Removes a loaded UID value from the cache.

Fails with an error if the UID does not exist, so be sure to check has_id beforehand.

---

void **set_id**(id: int, path: String) 🔗

Updates the resource path of an existing UID.

Fails with an error if the UID does not exist, so be sure to check has_id beforehand, or use add_id instead.

int **text_to_id**(text_id: String) const 🔗

Extracts the UID value from the given `uid://` string.

SceneMultiplayer

Inherits: MultiplayerAPI < RefCounted < Object

High-level multiplayer API implementation.

Description

This class is the default implementation of MultiplayerAPI, used to provide multiplayer functionalities in Godot Engine.

This implementation supports RPCs via Node.rpc and Node.rpc_id and requires MultiplayerAPI.rpc to be passed a Node (it will fail for other object types).

This implementation additionally provide SceneTree replication via the MultiplayerSpawner and MultiplayerSynchronizer nodes, and the SceneReplicationConfig resource.

Note: The high-level multiplayer API protocol is an implementation detail and isn't meant to be used by non-Godot servers. It may change without notice.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Properties

bool	allow_object_decoding	`false`

Callable	auth_callback	`Callable()`
float	auth_timeout	`3.0`
int	max_delta_packet_size	`65535`
int	max_sync_packet_size	`1350`
bool	refuse_new_connections	`false`
NodePath	root_path	`NodePath("")`
bool	server_relay	`true`

Methods

void	clear()
Error	complete_auth(id: int)
void	disconnect_peer(id: int)
PackedInt32Array	get_authenticating_peers()

Error	send_auth(id: int, data: PackedByteArray)
Error	send_bytes(bytes: PackedByteArray, id: int = 0, mode: TransferMode = 2, channel: int = 0)

Signals

peer_authenticating(id: int) 🔗

Emitted when this MultiplayerAPI's MultiplayerAPI.multiplayer_peer connects to a new peer and a valid auth_callback is set. In this case, the MultiplayerAPI.peer_connected will not be emitted until complete_auth is called with given peer `id`. While in this state, the peer will not be included in the list returned by MultiplayerAPI.get_peers (but in the one returned by get_authenticating_peers), and only authentication data will be sent or received. See send_auth for sending authentication data.

peer_authentication_failed(id: int) 🔗

Emitted when this MultiplayerAPI's MultiplayerAPI.multiplayer_peer disconnects from a peer for which authentication had not yet completed. See peer_authenticating.

peer_packet(id: int, packet: PackedByteArray) 🔗

Emitted when this MultiplayerAPI's MultiplayerAPI.multiplayer_peer receives a `packet` with custom data (see send_bytes). ID is the peer ID of the peer that sent the packet.

Property Descriptions

bool **allow_object_decoding** = `false` 🔗

- void **set_allow_object_decoding**(value: bool)
- bool **is_object_decoding_allowed**()

If `true`, the MultiplayerAPI will allow encoding and decoding of object during RPCs.

Warning: Deserialized objects can contain code which gets executed. Do not use this option if the serialized object comes from untrusted sources to avoid potential security threat such as remote code execution.

Callable **auth_callback** = `Callable()` 🔗

- void **set_auth_callback**(value: Callable)
- Callable **get_auth_callback**()

The callback to execute when when receiving authentication data sent via send_auth. If the Callable is empty (default), peers will be automatically accepted as soon as they connect.

---

float **auth_timeout** = `3.0` 🔗

- void **set_auth_timeout**(value: float)
- float **get_auth_timeout**()

If set to a value greater than `0.0`, the maximum amount of time peers can stay in the authenticating state, after which the authentication will automatically fail. See the peer_authenticating and peer_authentication_failed signals.

---

int **max_delta_packet_size** = `65535` 🔗

- void **set_max_delta_packet_size**(value: int)
- int **get_max_delta_packet_size**()

Maximum size of each delta packet. Higher values increase the chance of receiving full updates in a single frame, but also the chance of causing networking congestion (higher latency, disconnections). See MultiplayerSynchronizer.

int **max_sync_packet_size** = `1350` 🔗

- void **set_max_sync_packet_size**(value: int)
- int **get_max_sync_packet_size**()

Maximum size of each synchronization packet. Higher values increase the chance of receiving full updates in a single frame, but also the chance of packet loss. See MultiplayerSynchronizer.

bool **refuse_new_connections** = `false` 🔗

- void **set_refuse_new_connections**(value: bool)
- bool **is_refusing_new_connections**()

If `true`, the MultiplayerAPI's MultiplayerAPI.multiplayer_peer refuses new incoming connections.

NodePath **root_path** = `NodePath("")` 🔗

- void **set_root_path**(value: NodePath)
- NodePath **get_root_path**()

The root path to use for RPCs and replication. Instead of an absolute path, a relative path will be used to find the node upon which the

RPC should be executed.

This effectively allows to have different branches of the scene tree to be managed by different MultiplayerAPI, allowing for example to run both client and server in the same scene.

---

bool **server_relay** = `true`

- void **set_server_relay_enabled**(value: bool)
- bool **is_server_relay_enabled**()

Enable or disable the server feature that notifies clients of other peers' connection/disconnection, and relays messages between them. When this option is `false`, clients won't be automatically notified of other peers and won't be able to send them packets through the server.

Note: Changing this option while other peers are connected may lead to unexpected behaviors.

Note: Support for this feature may depend on the current MultiplayerPeer configuration. See MultiplayerPeer.is_server_relay_supported.

Method Descriptions

void **clear**()

Clears the current SceneMultiplayer network state (you shouldn't call this unless you know what you are doing).

Error **complete_auth**(id: int)

Mark the authentication step as completed for the remote peer identified by `id`. The MultiplayerAPI.peer_connected signal will be emitted for this peer once the remote side also completes the authentication. No further authentication messages are expected to be received from this peer.

If a peer disconnects before completing authentication, either due to a network issue, the auth_timeout expiring, or manually calling disconnect_peer, the peer_authentication_failed signal will be emitted instead of MultiplayerAPI.peer_disconnected.

void **disconnect_peer**(id: int) 🔗

Disconnects the peer identified by `id`, removing it from the list of connected peers, and closing the underlying connection with it.

PackedInt32Array **get_authenticating_peers**() 🔗

Returns the IDs of the peers currently trying to authenticate with this MultiplayerAPI.

Error **send_auth**(id: int, data: PackedByteArray) 🔗

Sends the specified `data` to the remote peer identified by `id` as part of an authentication message. This can be used to authenticate peers, and control when MultiplayerAPI.peer_connected is emitted (and the remote peer accepted as one of the connected peers).

Error **send_bytes**(bytes: PackedByteArray, id: int = 0, mode: TransferMode = 2, channel: int = 0) 🔗

Sends the given raw `bytes` to a specific peer identified by `id` (see MultiplayerPeer.set_target_peer). Default ID is `0`, i.e. broadcast to all peers.

SceneState

Inherits: RefCounted **<** Object

Provides access to a scene file's information.

Description

Maintains a list of resources, nodes, exported and overridden properties, and built-in scripts associated with a scene. They cannot be modified from a **SceneState**, only accessed. Useful for peeking into what a PackedScene contains without instantiating it.

This class cannot be instantiated directly, it is retrieved for a given scene as the result of PackedScene.get_state.

Methods

Array	get_connection_binds(idx: int) const
int	get_connection_count() const
int	get_connection_flags(idx: int) const
StringName	get_connection_method(idx: int) const
StringName	get_connection_signal(idx: int) const

NodePath	get_connection_source(idx: int) const
NodePath	get_connection_target(idx: int) const
int	get_connection_unbinds(idx: int) const
int	get_node_count() const
PackedStringArray	get_node_groups(idx: int) const
int	get_node_index(idx: int) const
PackedScene	get_node_instance(idx: int) const
String	get_node_instance_placeholder(idx: int) const
StringName	get_node_name(idx: int) const
NodePath	get_node_owner_path(idx: int) const
NodePath	get_node_path(idx: int, for_parent: bool = false) const
int	get_node_property_count(idx: int) const

StringName	get_node_property_name(idx: int, prop_idx: int) const
Variant	get_node_property_value(idx: int, prop_idx: int) const
StringName	get_node_type(idx: int) const
bool	is_node_instance_placeholder(idx: int) const

Enumerations

enum **GenEditState**: 🔗

GenEditState **GEN_EDIT_STATE_DISABLED** = `0`

If passed to PackedScene.instantiate, blocks edits to the scene state.

GenEditState **GEN_EDIT_STATE_INSTANCE** = `1`

If passed to PackedScene.instantiate, provides inherited scene resources to the local scene.

Note: Only available in editor builds.

GenEditState **GEN_EDIT_STATE_MAIN** = `2`

If passed to PackedScene.instantiate, provides local scene resources to the local scene. Only the main scene should receive the main edit state.

Note: Only available in editor builds.

GenEditState **GEN_EDIT_STATE_MAIN_INHERITED** = `3`

If passed to PackedScene.instantiate, it's similar to GEN_EDIT_STATE_MAIN, but for the case where the scene is being instantiated to be the base of another one.

Note: Only available in editor builds.

Method Descriptions

Array **get_connection_binds**(idx: int) const 🔗

Returns the list of bound parameters for the signal at `idx`.

---

int **get_connection_count**() const 🔗

Returns the number of signal connections in the scene.

The `idx` argument used to query connection metadata in other `get_connection_*` methods in the interval `[0, get_connection_count() - 1]`.

---

int **get_connection_flags**(idx: int) const 🔗

Returns the connection flags for the signal at `idx`. See ConnectFlags constants.

StringName **get_connection_method**(idx: int) const 🔗

Returns the method connected to the signal at `idx`.

StringName **get_connection_signal**(idx: int) const 🔗

Returns the name of the signal at `idx`.

NodePath **get_connection_source**(idx: int) const 🔗

Returns the path to the node that owns the signal at `idx`, relative to the root node.

NodePath **get_connection_target**(idx: int) const 🔗

Returns the path to the node that owns the method connected to the signal at `idx`, relative to the root node.

---

int **get_connection_unbinds**(idx: int) const 🔗

Returns the number of unbound parameters for the signal at `idx`.

---

int **get_node_count**() const 🔗

Returns the number of nodes in the scene.

The `idx` argument used to query node data in other `get_node_*` methods in the interval `[0, get_node_count() - 1]`.

PackedStringArray **get_node_groups**(idx: int) const 🔗

Returns the list of group names associated with the node at `idx`.

---

int **get_node_index**(idx: int) const 🔗

Returns the node's index, which is its position relative to its siblings. This is only relevant and saved in scenes for cases where new nodes are added to an instantiated or inherited scene among siblings from the base scene. Despite the name, this index is not related to the `idx` argument used here and in other methods.

PackedScene **get_node_instance**(idx: int) const 🔗

Returns a PackedScene for the node at `idx` (i.e. the whole branch starting at this node, with its child nodes and resources), or `null` if the node is not an instance.

String **get_node_instance_placeholder**(idx: int) const 🔗

Returns the path to the represented scene file if the node at `idx` is an InstancePlaceholder.

StringName **get_node_name**(idx: int) const 🔗

Returns the name of the node at `idx`.

NodePath **get_node_owner_path**(idx: int) const 🔗

Returns the path to the owner of the node at `idx`, relative to the root node.

NodePath **get_node_path**(idx: int, for_parent: bool = false) const 🔗

Returns the path to the node at `idx`.

If `for_parent` is `true`, returns the path of the `idx` node's parent instead.

---

int **get_node_property_count**(idx: int) const

Returns the number of exported or overridden properties for the node at `idx`.

The `prop_idx` argument used to query node property data in other `get_node_property_*` methods in the interval `[0, get_node_property_count() - 1]`.

StringName **get_node_property_name**(idx: int, prop_idx: int) const

Returns the name of the property at `prop_idx` for the node at `idx`.

Variant **get_node_property_value**(idx: int, prop_idx: int) const

Returns the value of the property at `prop_idx` for the node at `idx`.

StringName **get_node_type**(idx: int) const

Returns the type of the node at `idx`.

---

bool **is_node_instance_placeholder**(idx: int) const

Returns `true` if the node at `idx` is an InstancePlaceholder.

SceneTree

Inherits: MainLoop **<** Object

Manages the game loop via a hierarchy of nodes.

Description

As one of the most important classes, the **SceneTree** manages the hierarchy of nodes in a scene, as well as scenes themselves. Nodes can be added, fetched and removed. The whole scene tree (and thus the current scene) can be paused. Scenes can be loaded, switched and reloaded.

You can also use the **SceneTree** to organize your nodes into **groups**: every node can be added to as many groups as you want to create, e.g. an "enemy" group. You can then iterate these groups or even call methods and set properties on all the nodes belonging to any given group.

SceneTree is the default MainLoop implementation used by the engine, and is thus in charge of the game loop.

Tutorials

- SceneTree
- Multiple resolutions

Properties

bool	auto_accept_quit	`true`
Node	current_scene	
bool	debug_collisions_hint	`false`
bool	debug_navigation_hint	`false`
bool	debug_paths_hint	`false`
Node	edited_scene_root	
bool	multiplayer_poll	`true`
bool	paused	`false`
bool	physics_interpolation	`false`
bool	quit_on_go_back	`true`
Window	root	

Methods

void	call_group(group: StringName, method: StringName, ...) vararg
void	call_group_flags(flags: int, group: StringName, method: StringName, ...) vararg
Error	change_scene_to_file(path: String)
Error	change_scene_to_packed(packed_scene: PackedScene)
SceneTreeTimer	create_timer(time_sec: float, process_always: bool = true, process_in_physics: bool = false, ignore_time_scale: bool = false)
Tween	create_tween()
Node	get_first_node_in_group(group: StringName)
int	get_frame() const
MultiplayerAPI	get_multiplayer(for_path: NodePath = NodePath("")) const
int	get_node_count() const

int	get_node_count_in_group(group: StringName) const
Array[Node]	get_nodes_in_group(group: StringName)
Array[Tween]	get_processed_tweens()
bool	has_group(name: StringName) const
void	notify_group(group: StringName, notification: int)
void	notify_group_flags(call_flags: int, group: StringName, notification: int)
void	queue_delete(obj: Object)
void	quit(exit_code: int = 0)
Error	reload_current_scene()
void	set_group(group: StringName, property: String, value: Variant)
void	set_group_flags(call_flags: int, group: StringName, property: String, value: Variant)

void	set_multiplayer(multiplayer: MultiplayerAPI, root_path: NodePath = NodePath(""))
void	unload_current_scene()

Signals

node_added(node: Node) 🔗

Emitted when the `node` enters this tree.

---

node_configuration_warning_changed(node: Node) 🔗

Emitted when the `node`'s Node.update_configuration_warnings is called. Only emitted in the editor.

---

node_removed(node: Node) 🔗

Emitted when the `node` exits this tree.

---

node_renamed(node: Node) 🔗

Emitted when the `node`'s Node.name is changed.

---

physics_frame() 🔗

Emitted immediately before Node._physics_process is called on every node in this tree.

process_frame()

Emitted immediately before Node._process is called on every node in this tree.

tree_changed()

Emitted any time the tree's hierarchy changes (nodes being moved, renamed, etc.).

tree_process_mode_changed()

Emitted when the Node.process_mode of any node inside the tree is changed. Only emitted in the editor, to update the visibility of disabled nodes.

Enumerations

enum **GroupCallFlags**:

GroupCallFlags **GROUP_CALL_DEFAULT** = `0`

Call nodes within a group with no special behavior (default).

GroupCallFlags **GROUP_CALL_REVERSE** = `1`

Call nodes within a group in reverse tree hierarchy order (all nested children are called before their respective parent nodes).

GroupCallFlags **GROUP_CALL_DEFERRED** = `2`

Call nodes within a group at the end of the current frame (can be either process or physics frame), similar to Object.call_deferred.

GroupCallFlags **GROUP_CALL_UNIQUE** = `4`

Call nodes within a group only once, even if the call is executed many times in the same frame. Must be combined with GROUP_CALL_DEFERRED to work.

Note: Different arguments are not taken into account. Therefore, when the same call is executed with different arguments, only the first call will be performed.

Property Descriptions

bool **auto_accept_quit** = `true` 🔗

- void **set_auto_accept_quit**(value: bool)
- bool **is_auto_accept_quit**()

If `true`, the application automatically accepts quitting requests.

For mobile platforms, see quit_on_go_back.

Node **current_scene** 🔗

- void **set_current_scene**(value: Node)
- Node **get_current_scene**()

The root node of the currently loaded main scene, usually as a direct child of root. See also change_scene_to_file, change_scene_to_packed, and reload_current_scene.

Warning: Setting this property directly may not work as expected, as it does *not* add or remove any nodes from this tree.

---

bool **debug_collisions_hint** = `false`

- void **set_debug_collisions_hint**(value: bool)
- bool **is_debugging_collisions_hint**()

If `true`, collision shapes will be visible when running the game from the editor for debugging purposes.

Note: This property is not designed to be changed at run-time. Changing the value of debug_collisions_hint while the project is running will not have the desired effect.

---

bool **debug_navigation_hint** = `false`

- void **set_debug_navigation_hint**(value: bool)
- bool **is_debugging_navigation_hint**()

If `true`, navigation polygons will be visible when running the game from the editor for debugging purposes.

Note: This property is not designed to be changed at run-time. Changing the value of debug_navigation_hint while the project is running will not have the desired effect.

---

bool **debug_paths_hint** = `false`

- void **set_debug_paths_hint**(value: bool)
- bool **is_debugging_paths_hint**()

If `true`, curves from Path2D and Path3D nodes will be visible when running the game from the editor for debugging purposes.

Note: This property is not designed to be changed at run-time. Changing the value of debug_paths_hint while the project is running will not have the desired effect.

Node **edited_scene_root** 🔗

- void **set_edited_scene_root**(value: Node)
- Node **get_edited_scene_root**()

The root of the scene currently being edited in the editor. This is usually a direct child of root.

Note: This property does nothing in release builds.

bool **multiplayer_poll** = `true` 🔗

- void **set_multiplayer_poll_enabled**(value: bool)
- bool **is_multiplayer_poll_enabled**()

If `true` (default value), enables automatic polling of the MultiplayerAPI for this SceneTree during process_frame.

If `false`, you need to manually call MultiplayerAPI.poll to process network packets and deliver RPCs. This allows running RPCs in a different loop (e.g. physics, thread, specific time step) and for manual Mutex protection when accessing the MultiplayerAPI from threads.

bool **paused** = `false` 🔗

- void **set_pause**(value: bool)
- bool **is_paused**()

If `true`, the scene tree is considered paused. This causes the following behavior:

- 2D and 3D physics will be stopped, as well as collision detection and related signals.
- Depending on each node's Node.process_mode, their Node._process, Node._physics_process and Node._input callback methods may not called anymore.

---

bool **physics_interpolation** = `false` 🔗

- void **set_physics_interpolation_enabled**(value: bool)
- bool **is_physics_interpolation_enabled**()

If `true`, the renderer will interpolate the transforms of physics objects between the last two transforms, so that smooth motion is seen even when physics ticks do not coincide with rendered frames.

The default value of this property is controlled by ProjectSettings.physics/common/physics_interpolation.

---

bool **quit_on_go_back** = `true` 🔗

- void **set_quit_on_go_back**(value: bool)
- bool **is_quit_on_go_back**()

If `true`, the application quits automatically when navigating back (e.g. using the system "Back" button on Android).

To handle 'Go Back' button when this option is disabled, use DisplayServer.WINDOW_EVENT_GO_BACK_REQUEST.

Window **root** 🔗

- Window **get_root**()

The tree's root Window. This is top-most Node of the scene tree, and is always present. An absolute NodePath always starts from this node. Children of the root node may include the loaded current_scene, as well as any AutoLoad configured in the Project Settings.

Warning: Do not delete this node. This will result in unstable behavior, followed by a crash.

Method Descriptions

void **call_group**(group: StringName, method: StringName, ...) vararg 🔗

Calls `method` on each node inside this tree added to the given `group`. You can pass arguments to `method` by specifying them at the end of this method call. Nodes that cannot call `method` (either because the method doesn't exist or the arguments do not match) are ignored. See also set_group and notify_group.

Note: This method acts immediately on all selected nodes at once, which may cause stuttering in some performance-intensive situations.

Note: In C#, `method` must be in snake_case when referring to built-in Godot methods. Prefer using the names exposed in the `MethodName` class to avoid allocating a new StringName on each call.

---

void **call_group_flags**(flags: int, group: StringName, method: StringName, ...) vararg 🔗

Calls the given `method` on each node inside this tree added to the given `group`. Use `flags` to customize this method's behavior (see GroupCallFlags). Additional arguments for `method` can be passed at the end of this method. Nodes that cannot call `method` (either because the method doesn't exist or the arguments do not match) are ignored.

```
# Calls "hide" to all nodes of the "enemies" group, at the end of the frame and in reverse tree order.
get_tree().call_group_flags(
        SceneTree.GROUP_CALL_DEFERRED | SceneTree.GROUP_CALL_REVERSE,
        "enemies", "hide")
```

Note: In C#, `method` must be in snake_case when referring to built-in Godot methods. Prefer using the names exposed in the `MethodName` class to avoid allocating a new StringName on each call.

Error **change_scene_to_file**(path: String) 🔗

Changes the running scene to the one at the given `path`, after loading it into a PackedScene and creating a new instance.

Returns @GlobalScope.OK on success, @GlobalScope.ERR_CANT_OPEN if the `path` cannot be loaded into a PackedScene, or @GlobalScope.ERR_CANT_CREATE if that scene cannot be instantiated.

Note: See change_scene_to_packed for details on the order of operations.

Error **change_scene_to_packed**(packed_scene: PackedScene) 🔗

Changes the running scene to a new instance of the given PackedScene (which must be valid).

Returns @GlobalScope.OK on success, @GlobalScope.ERR_CANT_CREATE if the scene cannot be instantiated, or @GlobalScope.ERR_INVALID_PARAMETER if the scene is invalid.

Note: Operations happen in the following order when change_scene_to_packed is called:

1. The current scene node is immediately removed from the tree. From that point, Node.get_tree called on the current (outgoing) scene will return `null`. current_scene will be `null`, too, because the new scene is not available yet.
2. At the end of the frame, the formerly current scene, already removed from the tree, will be deleted (freed from memory) and then the new scene will be instantiated and added to the tree. Node.get_tree and current_scene will be back to working as usual.

This ensures that both scenes aren't running at the same time, while still freeing the previous scene in a safe way similar to Node.queue_free.

SceneTreeTimer **create_timer**(time_sec: float, process_always: bool = true, process_in_physics: bool = false, ignore_time_scale: bool = false) 🔗

Returns a new SceneTreeTimer. After `time_sec` in seconds have passed, the timer will emit SceneTreeTimer.timeout and will be automatically freed.

If `process_always` is `false`, the timer will be paused when setting paused to `true`.

If `process_in_physics` is `true`, the timer will update at the end of the physics frame, instead of the process frame.

If `ignore_time_scale` is `true`, the timer will ignore Engine.time_scale and update with the real, elapsed time.

This method is commonly used to create a one-shot delay timer, as in the following example:

GDScript

```
func some_function():
    print("start")
    await get_tree().create_timer(1.0).timeout
    print("end")
```

C#

```
public async Task SomeFunction()
{
    GD.Print("start");
    await ToSignal(GetTree().CreateTimer(1.0f),
SceneTreeTimer.SignalName.Timeout);
    GD.Print("end");
}
```

Note: The timer is always updated *after* all of the nodes in the tree. A node's Node._process method would be called before the timer updates (or Node._physics_process if `process_in_physics` is set to `true`).

Tween **create_tween**() 🔗

Creates and returns a new Tween processed in this tree. The Tween will start automatically on the next process frame or physics frame (depending on its TweenProcessMode).

Note: A Tween created using this method is not bound to any Node. It may keep working until there is nothing left to animate. If you want the Tween to be automatically killed when the Node is freed, use Node.create_tween or Tween.bind_node.

Node **get_first_node_in_group**(group: StringName) 🔗

Returns the first Node found inside the tree, that has been added to the given `group`, in scene hierarchy order. Returns `null` if no match is found. See also get_nodes_in_group.

---

int **get_frame**() const 🔗

Returns how many frames have been processed, since the application started. This is *not* a measurement of elapsed time.

MultiplayerAPI **get_multiplayer**(for_path: NodePath = NodePath("")) const 🔗

Searches for the MultiplayerAPI configured for the given path, if one does not exist it searches the parent paths until one is found. If the path is empty, or none is found, the default one is returned. See set_multiplayer.

---

int **get_node_count**() const 🔗

Returns the number of nodes inside this tree.

---

int **get_node_count_in_group**(group: StringName) const 🔗

Returns the number of nodes assigned to the given group.

Array[Node] **get_nodes_in_group**(group: StringName) 🔗

Returns an Array containing all nodes inside this tree, that have been added to the given `group`, in scene hierarchy order.

Array[Tween] **get_processed_tweens**() 🔗

Returns an Array of currently existing Tweens in the tree, including paused tweens.

---

bool **has_group**(name: StringName) const 🔗

Returns `true` if a node added to the given group `name` exists in the tree.

---

void **notify_group**(group: StringName, notification: int) 🔗

Calls Object.notification with the given `notification` to all nodes inside this tree added to the `group`. See also Godot notifications and call_group and set_group.

Note: This method acts immediately on all selected nodes at once, which may cause stuttering in some performance-intensive situations.

---

void **notify_group_flags**(call_flags: int, group: StringName, notification: int) 🔗

Calls Object.notification with the given `notification` to all nodes inside this tree added to the `group`. Use `call_flags` to customize this method's behavior (see GroupCallFlags).

---

void **queue_delete**(obj: Object) 🔗

Queues the given `obj` to be deleted, calling its Object.free at the end of the current frame. This method is similar to Node.queue_free.

---

void **quit**(exit_code: int = 0) 🔗

Quits the application at the end of the current iteration, with the given `exit_code`.

By convention, an exit code of `0` indicates success, whereas any other exit code indicates an error. For portability reasons, it should be between `0` and `125` (inclusive).

Note: On iOS this method doesn't work. Instead, as recommended by the iOS Human Interface Guidelines [https://developer.apple.com/library/archive/qa/qa1561/_index.html], the user is expected to close apps via the Home button.

Error **reload_current_scene**() 🔗

Reloads the currently active scene, replacing current_scene with a new instance of its original PackedScene.

Returns @GlobalScope.OK on success, @GlobalScope.ERR_UNCONFIGURED if no current_scene is defined, @GlobalScope.ERR_CANT_OPEN if current_scene cannot be loaded into a PackedScene, or @GlobalScope.ERR_CANT_CREATE if the scene cannot be instantiated.

---

void **set_group**(group: StringName, property: String, value: Variant) 🔗

Sets the given `property` to `value` on all nodes inside this tree added to the given `group`. Nodes that do not have the `property` are ignored. See also call_group and notify_group.

Note: This method acts immediately on all selected nodes at once, which may cause stuttering in some performance-intensive situations.

Note: In C#, `property` must be in snake_case when referring to built-in Godot properties. Prefer using the names exposed in the `PropertyName` class to avoid allocating a new StringName on each call.

---

void **set_group_flags**(call_flags: int, group: StringName, property: String, value: Variant)

Sets the given `property` to `value` on all nodes inside this tree added to the given `group`. Nodes that do not have the `property` are ignored. Use `call_flags` to customize this method's behavior (see GroupCallFlags).

Note: In C#, `property` must be in snake_case when referring to built-in Godot properties. Prefer using the names exposed in the `PropertyName` class to avoid allocating a new StringName on each call.

---

void **set_multiplayer**(multiplayer: MultiplayerAPI, root_path: NodePath = NodePath(""))

Sets a custom MultiplayerAPI with the given `root_path` (controlling also the relative subpaths), or override the default one if `root_path` is empty.

Note: No MultiplayerAPI must be configured for the subpath containing `root_path`, nested custom multiplayers are not allowed. I.e. if one is configured for `"/root/Foo"` setting one for `"/root/Foo/Bar"` will cause an error.

---

void **unload_current_scene**() 🔗

If a current scene is loaded, calling this method will unload it.

SceneTreeTimer

Inherits: RefCounted < Object

One-shot timer.

Description

A one-shot timer managed by the scene tree, which emits timeout on completion. See also SceneTree.create_timer.

As opposed to Timer, it does not require the instantiation of a node. Commonly used to create a one-shot delay timer as in the following example:

GDScript

```
func some_function():
    print("Timer started.")
    await get_tree().create_timer(1.0).timeout
    print("Timer ended.")
```

C#

```
public async Task SomeFunction()
{
    GD.Print("Timer started.");
    await ToSignal(GetTree().CreateTimer(1.0f),
SceneTreeTimer.SignalName.Timeout);
    GD.Print("Timer ended.");
}
```

The timer will be dereferenced after its time elapses. To preserve the timer, you can keep a reference to it. See RefCounted.

Note: The timer is processed after all of the nodes in the current frame, i.e. node's Node._process method would be called before the timer (or Node._physics_process if `process_in_physics` in SceneTree.create_timer has been set to `true`).

Properties

float	time_left

Signals

timeout()

Emitted when the timer reaches 0.

Property Descriptions

float **time_left**

- void **set_time_left**(value: float)
- float **get_time_left**()

The time remaining (in seconds).

ScriptLanguage

Inherits: Object

Inherited By: ScriptLanguageExtension

There is currently no description for this class. Please help us by contributing one!

Enumerations

enum **ScriptNameCasing**: 🔗

ScriptNameCasing **SCRIPT_NAME_CASING_AUTO** = `0`

There is currently no description for this enum. Please help us by contributing one!

ScriptNameCasing **SCRIPT_NAME_CASING_PASCAL_CASE** = `1`

There is currently no description for this enum. Please help us by contributing one!

ScriptNameCasing **SCRIPT_NAME_CASING_SNAKE_CASE** = `2`

There is currently no description for this enum. Please help us by contributing one!

ScriptNameCasing **SCRIPT_NAME_CASING_KEBAB_CASE** = `3`

There is currently no description for this enum. Please help us by contributing one!

ScriptLanguageExtension

Inherits: ScriptLanguage **<** Object

There is currently no description for this class. Please help us by contributing one!

Methods

void	_add_global_constant(name: StringName, value: Variant) virtual
void	_add_named_global_constant(name: StringName, value: Variant) virtual
String	_auto_indent_code(code: String, from_line: int, to_line: int) virtual const
bool	_can_inherit_from_file() virtual const
bool	_can_make_function() virtual const
Dictionary	_complete_code(code: String, path: String, owner: Object) virtual const
Object	_create_script() virtual const

Array[Dictionary]	_debug_get_current_stack_info() virtual
String	_debug_get_error() virtual const
Dictionary	_debug_get_globals(max_subitems: int, max_depth: int) virtual
int	_debug_get_stack_level_count() virtual const
String	_debug_get_stack_level_function(level: int) virtual const
void*	_debug_get_stack_level_instance(level: int) virtual
int	_debug_get_stack_level_line(level: int) virtual const
Dictionary	_debug_get_stack_level_locals(level: int, max_subitems: int, max_depth: int) virtual
Dictionary	_debug_get_stack_level_members(level: int, max_subitems: int, max_depth: int) virtual
String	_debug_get_stack_level_source(level: int) virtual const

String	_debug_parse_stack_level_expression(level: int, expression: String, max_subitems: int, max_depth: int) virtual
int	_find_function(function: String, code: String) virtual const
void	_finish() virtual
void	_frame() virtual
Array[Dictionary]	_get_built_in_templates(object: StringName) virtual const
PackedStringArray	_get_comment_delimiters() virtual const
PackedStringArray	_get_doc_comment_delimiters() virtual const
String	_get_extension() virtual const
Dictionary	_get_global_class_name(path: String) virtual const
String	_get_name() virtual const
Array[Dictionary]	_get_public_annotations() virtual const

Dictionary	_get_public_constants() virtual const
Array[Dictionary]	_get_public_functions() virtual const
PackedStringArray	_get_recognized_extensions() virtual const
PackedStringArray	_get_reserved_words() virtual const
PackedStringArray	_get_string_delimiters() virtual const
String	_get_type() virtual const
bool	_handles_global_class_type(type: String) virtual const
bool	_has_named_classes() virtual const
void	_init() virtual
bool	_is_control_flow_keyword(keyword: String) virtual const
bool	_is_using_templates() virtual
Dictionary	_lookup_code(code: String, symbol: String, path: String, owner: Object) virtual const

String	_make_function(class_name: String, function_name: String, function_args: PackedStringArray) virtual const
Script	_make_template(template: String, class_name: String, base_class_name: String) virtual const
Error	_open_in_external_editor(script: Script, line: int, column: int) virtual
bool	_overrides_external_editor() virtual
ScriptNameCasing	_preferred_file_name_casing() virtual const
int	_profiling_get_accumulated_data(info_array: `ScriptLanguageExtensionProfilingInfo*`, info_max: int) virtual
int	_profiling_get_frame_data(info_array: `ScriptLanguageExtensionProfilingInfo*`, info_max: int) virtual
void	_profiling_set_save_native_calls(enable: bool) virtual
void	_profiling_start() virtual

void	_profiling_stop() virtual
void	_reload_all_scripts() virtual
void	_reload_scripts(scripts: Array, soft_reload: bool) virtual
void	_reload_tool_script(script: Script, soft_reload: bool) virtual
void	_remove_named_global_constant(name: StringName) virtual
bool	_supports_builtin_mode() virtual const
bool	_supports_documentation() virtual const
void	_thread_enter() virtual
void	_thread_exit() virtual
Dictionary	_validate(script: String, path: String, validate_functions: bool, validate_errors: bool, validate_warnings: bool, validate_safe_lines: bool) virtual const
String	_validate_path(path: String) virtual const

Enumerations

enum **LookupResultType**:

LookupResultType **LOOKUP_RESULT_SCRIPT_LOCATION** = `0`

There is currently no description for this enum. Please help us by contributing one!

LookupResultType **LOOKUP_RESULT_CLASS** = `1`

There is currently no description for this enum. Please help us by contributing one!

LookupResultType **LOOKUP_RESULT_CLASS_CONSTANT** = `2`

There is currently no description for this enum. Please help us by contributing one!

LookupResultType **LOOKUP_RESULT_CLASS_PROPERTY** = `3`

There is currently no description for this enum. Please help us by contributing one!

LookupResultType **LOOKUP_RESULT_CLASS_METHOD** = `4`

There is currently no description for this enum. Please help us by contributing one!

LookupResultType **LOOKUP_RESULT_CLASS_SIGNAL** = `5`

There is currently no description for this enum. Please help us by contributing one!

LookupResultType **LOOKUP_RESULT_CLASS_ENUM** = `6`

There is currently no description for this enum. Please help us by contributing one!

LookupResultType **LOOKUP_RESULT_CLASS_TBD_GLOBALSCOPE** = `7`

There is currently no description for this enum. Please help us by contributing one!

LookupResultType **LOOKUP_RESULT_CLASS_ANNOTATION** = `8`

There is currently no description for this enum. Please help us by contributing one!

LookupResultType **LOOKUP_RESULT_MAX** = `9`

There is currently no description for this enum. Please help us by contributing one!

---

enum **CodeCompletionLocation**: 🔗

CodeCompletionLocation **LOCATION_LOCAL** = `0`

The option is local to the location of the code completion query - e.g. a local variable. Subsequent value of location represent options from the outer class, the exact value represent how far they are (in terms of inner classes).

CodeCompletionLocation **LOCATION_PARENT_MASK** = `256`

The option is from the containing class or a parent class, relative to the location of the code completion query. Perform a bitwise OR with the class depth (e.g. `0` for the local class, `1` for the parent, `2` for the grandparent, etc.) to store the depth of an option in the class or a parent class.

CodeCompletionLocation **LOCATION_OTHER_USER_CODE** = `512`

The option is from user code which is not local and not in a derived class (e.g. Autoload Singletons).

CodeCompletionLocation **LOCATION_OTHER** = `1024`

The option is from other engine code, not covered by the other enum constants - e.g. built-in classes.

---

enum **CodeCompletionKind**: 🔗

CodeCompletionKind **CODE_COMPLETION_KIND_CLASS** = `0`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_FUNCTION** = `1`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_SIGNAL** = `2`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_VARIABLE** = `3`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_MEMBER** = `4`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_ENUM** = `5`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_CONSTANT** = `6`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_NODE_PATH** = `7`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_FILE_PATH** = `8`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_PLAIN_TEXT** = `9`

There is currently no description for this enum. Please help us by contributing one!

CodeCompletionKind **CODE_COMPLETION_KIND_MAX** = `10`

There is currently no description for this enum. Please help us by contributing one!

Method Descriptions

void **_add_global_constant**(name: StringName, value: Variant) virtual

There is currently no description for this method. Please help us by contributing one!

void **_add_named_global_constant**(name: StringName, value: Variant) virtual

There is currently no description for this method. Please help us by contributing one!

String **_auto_indent_code**(code: String, from_line: int, to_line: int) virtual const

There is currently no description for this method. Please help us by contributing one!

bool **_can_inherit_from_file**() virtual const

There is currently no description for this method. Please help us by contributing one!

bool **_can_make_function**() virtual const

There is currently no description for this method. Please help us by contributing one!

Dictionary **_complete_code**(code: String, path: String, owner: Object) virtual const

There is currently no description for this method. Please help us by contributing one!

Object **_create_script**() virtual const

There is currently no description for this method. Please help us by contributing one!

Array[Dictionary] **_debug_get_current_stack_info**() virtual

There is currently no description for this method. Please help us by contributing one!

String **_debug_get_error**() virtual const

There is currently no description for this method. Please help us by contributing one!

Dictionary **_debug_get_globals**(max_subitems: int, max_depth: int) virtual

There is currently no description for this method. Please help us by contributing one!

---

int **_debug_get_stack_level_count**() virtual const

There is currently no description for this method. Please help us by contributing one!

String **_debug_get_stack_level_function**(level: int) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

`void*` **_debug_get_stack_level_instance**(level: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

int **_debug_get_stack_level_line**(level: int) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Dictionary **_debug_get_stack_level_locals**(level: int, max_subitems: int, max_depth: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

Dictionary **_debug_get_stack_level_members**(level: int, max_subitems: int, max_depth: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

String **_debug_get_stack_level_source**(level: int) virtual const 🔗

Returns the source associated with a given debug stack position.

String **_debug_parse_stack_level_expression**(level: int, expression: String, max_subitems: int, max_depth: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_find_function**(function: String, code: String) virtual const 🔗

Returns the line where the function is defined in the code, or `-1` if the function is not present.

---

void **_finish**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_frame**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

Array[Dictionary] **_get_built_in_templates**(object: StringName) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

PackedStringArray **_get_comment_delimiters**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

PackedStringArray **_get_doc_comment_delimiters**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

String **_get_extension**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Dictionary **_get_global_class_name**(path: String) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

String **_get_name**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Array[Dictionary] **_get_public_annotations**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Dictionary **_get_public_constants**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Array[Dictionary] **_get_public_functions**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

PackedStringArray **_get_recognized_extensions**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

PackedStringArray **_get_reserved_words**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

PackedStringArray **_get_string_delimiters**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

String **_get_type**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_handles_global_class_type**(type: String) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_has_named_classes**() virtual const 🔗

Deprecated: This method is not called by the engine.

---

void **_init**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_is_control_flow_keyword**(keyword: String) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_is_using_templates**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

Dictionary **_lookup_code**(code: String, symbol: String, path: String, owner: Object) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

String **_make_function**(class_name: String, function_name: String, function_args: PackedStringArray) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Script **_make_template**(template: String, class_name: String, base_class_name: String) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Error **_open_in_external_editor**(script: Script, line: int, column: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

bool **_overrides_external_editor**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

ScriptNameCasing **_preferred_file_name_casing**() virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_profiling_get_accumulated_data**(info_array: `ScriptLanguageExtensionProfilingInfo*`, info_max: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

int **_profiling_get_frame_data**(info_array: `ScriptLanguageExtensionProfilingInfo*`, info_max: int) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_profiling_set_save_native_calls**(enable: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_profiling_start**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_profiling_stop**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_reload_all_scripts**() virtual 🔗

There is currently no description for this method. Please help us by contributing one!

---

void **_reload_scripts**(scripts: Array, soft_reload: bool) virtual 🔗

There is currently no description for this method. Please help us by contributing one!

void **_reload_tool_script**(script: Script, soft_reload: bool) virtual

There is currently no description for this method. Please help us by contributing one!

void **_remove_named_global_constant**(name: StringName) virtual

There is currently no description for this method. Please help us by contributing one!

bool **_supports_builtin_mode**() virtual const

There is currently no description for this method. Please help us by contributing one!

bool **_supports_documentation**() virtual const

There is currently no description for this method. Please help us by contributing one!

void **_thread_enter**() virtual

There is currently no description for this method. Please help us by contributing one!

void **_thread_exit**() virtual

There is currently no description for this method. Please help us by contributing one!

Dictionary **_validate**(script: String, path: String, validate_functions: bool, validate_errors: bool, validate_warnings: bool, validate_safe_lines: bool) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

String **_validate_path**(path: String) virtual const 🔗

There is currently no description for this method. Please help us by contributing one!

Semaphore

Inherits: RefCounted **<** Object

A synchronization mechanism used to control access to a shared resource by Threads.

Description

A synchronization semaphore that can be used to synchronize multiple Threads. Initialized to zero on creation. For a binary version, see Mutex.

Warning: Semaphores must be used carefully to avoid deadlocks.

Warning: To guarantee that the operating system is able to perform proper cleanup (no crashes, no deadlocks), these conditions must be met:

- When a **Semaphore**'s reference count reaches zero and it is therefore destroyed, no threads must be waiting on it.
- When a Thread's reference count reaches zero and it is therefore destroyed, it must not be waiting on any semaphore.

Tutorials

- Using multiple threads
- Thread-safe APIs

Methods

void	post()
bool	try_wait()
void	wait()

Method Descriptions

void **post**() 🔗

Lowers the **Semaphore**, allowing one more thread in.

bool **try_wait**() 🔗

Like wait, but won't block, so if the value is zero, fails immediately and returns `false`. If non-zero, it returns `true` to report success.

void **wait**() 🔗

Waits for the **Semaphore**, if its value is zero, blocks until non-zero.

SkinReference

Inherits: RefCounted < Object

A reference-counted holder object for a skeleton RID used in the RenderingServer.

Description

An internal object containing a mapping from a Skin used within the context of a particular MeshInstance3D to refer to the skeleton's RID in the RenderingServer.

See also MeshInstance3D.get_skin_reference and RenderingServer.instance_attach_skeleton.

Note that despite the similar naming, the skeleton RID used in the RenderingServer does not have a direct one-to-one correspondence to a Skeleton3D node.

In particular, a Skeleton3D node with no MeshInstance3D children may be unknown to the RenderingServer.

On the other hand, a Skeleton3D with multiple MeshInstance3D nodes which each have different MeshInstance3D.skin objects may have multiple SkinReference instances (and hence, multiple skeleton RIDs).

Methods

RID	get_skeleton() const

Skin	get_skin() const

Method Descriptions

RID **get_skeleton**() const 🔗

Returns the RID owned by this SkinReference, as returned by RenderingServer.skeleton_create.

Skin **get_skin**() const 🔗

Returns the Skin connected to this SkinReference. In the case of MeshInstance3D with no MeshInstance3D.skin assigned, this will reference an internal default Skin owned by that MeshInstance3D.

Note that a single Skin may have more than one **SkinReference** in the case that it is shared by meshes across multiple Skeleton3D nodes.

StreamPeer

Inherits: RefCounted **<** Object

Inherited By: StreamPeerBuffer, StreamPeerExtension, StreamPeerGZIP, StreamPeerTCP, StreamPeerTLS

Abstract base class for interacting with streams.

Description

StreamPeer is an abstract base class mostly used for stream-based protocols (such as TCP). It provides an API for sending and receiving data through streams as raw data or strings.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Properties

bool	big_endian	`false`

Methods

int	get_8()

int	get_16()
int	get_32()
int	get_64()
int	get_available_bytes() const
Array	get_data(bytes: int)
float	get_double()
float	get_float()
Array	get_partial_data(bytes: int)
String	get_string(bytes: int = -1)
int	get_u8()
int	get_u16()
int	get_u32()
int	get_u64()

String	get_utf8_string(bytes: int = -1)
Variant	get_var(allow_objects: bool = false)
void	put_8(value: int)
void	put_16(value: int)
void	put_32(value: int)
void	put_64(value: int)
Error	put_data(data: PackedByteArray)
void	put_double(value: float)
void	put_float(value: float)
Array	put_partial_data(data: PackedByteArray)
void	put_string(value: String)
void	put_u8(value: int)
void	put_u16(value: int)

void	put_u32(value: int)
void	put_u64(value: int)
void	put_utf8_string(value: String)
void	put_var(value: Variant, full_objects: bool = false)

Property Descriptions

bool **big_endian** = `false`

- void **set_big_endian**(value: bool)
- bool **is_big_endian_enabled**()

If `true`, this **StreamPeer** will using big-endian format for encoding and decoding.

Method Descriptions

int **get_8**()

Gets a signed byte from the stream.

int **get_16**()

Gets a signed 16-bit value from the stream.

int **get_32**()

Gets a signed 32-bit value from the stream.

int **get_64**()

Gets a signed 64-bit value from the stream.

int **get_available_bytes**() const

Returns the number of bytes this **StreamPeer** has available.

Array **get_data**(bytes: int)

Returns a chunk data with the received bytes. The number of bytes to be received can be requested in the `bytes` argument. If not enough bytes are available, the function will block until the desired amount is received. This function returns two values, an Error code and a data array.

float **get_double**()

Gets a double-precision float from the stream.

float **get_float**()

Gets a single-precision float from the stream.

Array **get_partial_data**(bytes: int)

Returns a chunk data with the received bytes. The number of bytes to be received can be requested in the "bytes" argument. If not enough bytes are available, the function will return how many were actually received. This function returns two values, an Error code, and a data array.

String **get_string**(bytes: int = -1)

Gets an ASCII string with byte-length `bytes` from the stream. If `bytes` is negative (default) the length will be read from the stream using the reverse process of put_string.

int **get_u8**()

Gets an unsigned byte from the stream.

int **get_u16**()

Gets an unsigned 16-bit value from the stream.

int **get_u32**()

Gets an unsigned 32-bit value from the stream.

int **get_u64**()

Gets an unsigned 64-bit value from the stream.

String **get_utf8_string**(bytes: int = -1)

Gets a UTF-8 string with byte-length `bytes` from the stream (this decodes the string sent as UTF-8). If `bytes` is negative (default) the length will be read from the stream using the reverse process of put_utf8_string.

Variant **get_var**(allow_objects: bool = false)

Gets a Variant from the stream. If `allow_objects` is `true`, decoding objects is allowed.

Internally, this uses the same decoding mechanism as the @GlobalScope.bytes_to_var method.

Warning: Deserialized objects can contain code which gets executed. Do not use this option if the serialized object comes from untrusted sources to avoid potential security threats such as remote code execution.

---

void **put_8**(value: int)

Puts a signed byte into the stream.

---

void **put_16**(value: int)

Puts a signed 16-bit value into the stream.

---

void **put_32**(value: int)

Puts a signed 32-bit value into the stream.

---

void **put_64**(value: int)

Puts a signed 64-bit value into the stream.

Error **put_data**(data: PackedByteArray) 🔗

Sends a chunk of data through the connection, blocking if necessary until the data is done sending. This function returns an Error code.

---

void **put_double**(value: float) 🔗

Puts a double-precision float into the stream.

---

void **put_float**(value: float) 🔗

Puts a single-precision float into the stream.

Array **put_partial_data**(data: PackedByteArray) 🔗

Sends a chunk of data through the connection. If all the data could not be sent at once, only part of it will. This function returns two values, an Error code and an integer, describing how much data was actually sent.

---

void **put_string**(value: String) 🔗

Puts a zero-terminated ASCII string into the stream prepended by a 32-bit unsigned integer representing its size.

Note: To put an ASCII string without prepending its size, you can use put_data:

GDScript

```
put_data("Hello world".to_ascii_buffer())
```

C#

```
PutData("Hello World".ToAsciiBuffer());
```

---

void **put_u8**(value: int) 🔗

Puts an unsigned byte into the stream.

---

void **put_u16**(value: int) 🔗

Puts an unsigned 16-bit value into the stream.

---

void **put_u32**(value: int) 🔗

Puts an unsigned 32-bit value into the stream.

---

void **put_u64**(value: int) 🔗

Puts an unsigned 64-bit value into the stream.

---

void **put_utf8_string**(value: String) 🔗

Puts a zero-terminated UTF-8 string into the stream prepended by a 32 bits unsigned integer representing its size.

Note: To put a UTF-8 string without prepending its size, you can use put_data:

GDScript

```
put_data("Hello world".to_utf8_buffer())
```

C#

```
PutData("Hello World".ToUtf8Buffer());
```

---

void **put_var**(value: Variant, full_objects: bool = false) 🔗

Puts a Variant into the stream. If `full_objects` is `true` encoding objects is allowed (and can potentially include code).

Internally, this uses the same encoding mechanism as the @GlobalScope.var_to_bytes method.

StreamPeerBuffer

Inherits: StreamPeer **<** RefCounted **<** Object

A stream peer used to handle binary data streams.

Description

A data buffer stream peer that uses a byte array as the stream. This object can be used to handle binary data from network sessions. To handle binary data stored in files, FileAccess can be used directly.

A **StreamPeerBuffer** object keeps an internal cursor which is the offset in bytes to the start of the buffer. Get and put operations are performed at the cursor position and will move the cursor accordingly.

Properties

PackedByteArray	data_array	`PackedByteArray()`

Methods

void	clear()
StreamPeerBuffer	duplicate() const

int	get_position() const
int	get_size() const
void	resize(size: int)
void	seek(position: int)

Property Descriptions

PackedByteArray **data_array** = `PackedByteArray()`

- void **set_data_array**(value: PackedByteArray)
- PackedByteArray **get_data_array**()

The underlying data buffer. Setting this value resets the cursor.

Note: The returned array is *copied* and any changes to it will not update the original property value. See PackedByteArray for more details.

Method Descriptions

void **clear**()

Clears the data_array and resets the cursor.

StreamPeerBuffer **duplicate**() const

Returns a new **StreamPeerBuffer** with the same data_array content.

int **get_position**() const

Returns the current cursor position.

int **get_size**() const

Returns the size of data_array.

void **resize**(size: int)

Resizes the data_array. This *doesn't* update the cursor.

void **seek**(position: int)

Moves the cursor to the specified position. `position` must be a valid index of data_array.

StreamPeerExtension

Inherits: StreamPeer **<** RefCounted **<** Object

There is currently no description for this class. Please help us by contributing one!

Methods

int	_get_available_bytes() virtual const
Error	_get_data(r_buffer: `uint8_t*`, r_bytes: int, r_received: `int32_t*`) virtual
Error	_get_partial_data(r_buffer: `uint8_t*`, r_bytes: int, r_received: `int32_t*`) virtual
Error	_put_data(p_data: `const uint8_t*`, p_bytes: int, r_sent: `int32_t*`) virtual
Error	_put_partial_data(p_data: `const uint8_t*`, p_bytes: int, r_sent: `int32_t*`) virtual

Method Descriptions

int **_get_available_bytes**() virtual const

There is currently no description for this method. Please help us by contributing one!

Error **_get_data**(r_buffer: `uint8_t*`, r_bytes: int, r_received: `int32_t*`) virtual

There is currently no description for this method. Please help us by contributing one!

Error **_get_partial_data**(r_buffer: `uint8_t*`, r_bytes: int, r_received: `int32_t*`) virtual

There is currently no description for this method. Please help us by contributing one!

Error **_put_data**(p_data: `const uint8_t*`, p_bytes: int, r_sent: `int32_t*`) virtual

There is currently no description for this method. Please help us by contributing one!

Error **_put_partial_data**(p_data: `const uint8_t*`, p_bytes: int, r_sent: `int32_t*`) virtual

There is currently no description for this method. Please help us by contributing one!

StreamPeerGZIP

Experimental: This class may be changed or removed in future versions.

Inherits: StreamPeer **<** RefCounted **<** Object

A stream peer that handles GZIP and deflate compression/decompression.

Description

This class allows to compress or decompress data using GZIP/deflate in a streaming fashion. This is particularly useful when compressing or decompressing files that have to be sent through the network without needing to allocate them all in memory.

After starting the stream via start_compression (or start_decompression), calling StreamPeer.put_partial_data on this stream will compress (or decompress) the data, writing it to the internal buffer. Calling StreamPeer.get_available_bytes will return the pending bytes in the internal buffer, and StreamPeer.get_partial_data will retrieve the compressed (or decompressed) bytes from it. When the stream is over, you must call finish to ensure the internal buffer is properly flushed (make sure to call StreamPeer.get_available_bytes on last time to check if more data needs to be read after that).

Methods

void	clear()

Error	finish()
Error	start_compression(use_deflate: bool = false, buffer_size: int = 65535)
Error	start_decompression(use_deflate: bool = false, buffer_size: int = 65535)

Method Descriptions

void **clear**() 🔗

Clears this stream, resetting the internal state.

Error **finish**() 🔗

Finalizes the stream, compressing or decompressing any buffered chunk left.

Error **start_compression**(use_deflate: bool = false, buffer_size: int = 65535) 🔗

Start the stream in compression mode with the given `buffer_size`, if `use_deflate` is `true` uses deflate instead of GZIP.

Error **start_decompression**(use_deflate: bool = false, buffer_size: int = 65535) 🔗

Start the stream in decompression mode with the given `buffer_size`, if `use_deflate` is `true` uses deflate instead of GZIP.

StreamPeerTCP

Inherits: StreamPeer **<** RefCounted **<** Object

A stream peer that handles TCP connections.

Description

A stream peer that handles TCP connections. This object can be used to connect to TCP servers, or also is returned by a TCP server.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Methods

Error	bind(port: int, host: String = "*")
Error	connect_to_host(host: String, port: int)
void	disconnect_from_host()
String	get_connected_host() const
int	get_connected_port() const

int	get_local_port() const
Status	get_status() const
Error	poll()
void	set_no_delay(enabled: bool)

Enumerations

enum **Status**: 🔗

Status **STATUS_NONE** = `0`

The initial status of the **StreamPeerTCP**. This is also the status after disconnecting.

Status **STATUS_CONNECTING** = `1`

A status representing a **StreamPeerTCP** that is connecting to a host.

Status **STATUS_CONNECTED** = `2`

A status representing a **StreamPeerTCP** that is connected to a host.

Status **STATUS_ERROR** = `3`

A status representing a **StreamPeerTCP** in error state.

Method Descriptions

Error **bind**(port: int, host: String = "*")

Opens the TCP socket, and binds it to the specified local address.

This method is generally not needed, and only used to force the subsequent call to connect_to_host to use the specified `host` and `port` as source address. This can be desired in some NAT punchthrough techniques, or when forcing the source network interface.

Error **connect_to_host**(host: String, port: int)

Connects to the specified `host:port` pair. A hostname will be resolved if valid. Returns @GlobalScope.OK on success.

---

void **disconnect_from_host**()

Disconnects from host.

String **get_connected_host**() const

Returns the IP of this peer.

---

int **get_connected_port**() const

Returns the port of this peer.

---

int **get_local_port**() const

Returns the local port to which this peer is bound.

Status **get_status**() const

Returns the status of the connection, see Status.

Error **poll**()

Poll the socket, updating its state. See get_status.

---

void **set_no_delay**(enabled: bool)

If `enabled` is `true`, packets will be sent immediately. If `enabled` is `false` (the default), packet transfers will be delayed and combined using Nagle's algorithm [https://en.wikipedia.org/wiki/Nagle%27s_algorithm].

Note: It's recommended to leave this disabled for applications that send large packets or need to transfer a lot of data, as enabling this can decrease the total available bandwidth.

StreamPeerTLS

Inherits: StreamPeer **<** RefCounted **<** Object

A stream peer that handles TLS connections.

Description

A stream peer that handles TLS connections. This object can be used to connect to a TLS server or accept a single TLS client connection.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Tutorials

- TLS certificates

Methods

Error	accept_stream(stream: StreamPeer, server_options: TLSOptions)
Error	connect_to_stream(stream: StreamPeer, common_name: String, client_options: TLSOptions = null)

void	disconnect_from_stream()
Status	get_status() const
StreamPeer	get_stream() const
void	poll()

Enumerations

enum **Status**: 🔗

Status **STATUS_DISCONNECTED** = `0`

A status representing a **StreamPeerTLS** that is disconnected.

Status **STATUS_HANDSHAKING** = `1`

A status representing a **StreamPeerTLS** during handshaking.

Status **STATUS_CONNECTED** = `2`

A status representing a **StreamPeerTLS** that is connected to a host.

Status **STATUS_ERROR** = `3`

A status representing a **StreamPeerTLS** in error state.

Status **STATUS_ERROR_HOSTNAME_MISMATCH** = `4`

An error status that shows a mismatch in the TLS certificate domain presented by the host and the domain requested for validation.

Method Descriptions

Error **accept_stream**(stream: StreamPeer, server_options: TLSOptions) 🔗

Accepts a peer connection as a server using the given `server_options`. See TLSOptions.server.

Error **connect_to_stream**(stream: StreamPeer, common_name: String, client_options: TLSOptions = null) 🔗

Connects to a peer using an underlying StreamPeer `stream` and verifying the remote certificate is correctly signed for the given `common_name`. You can pass the optional `client_options` parameter to customize the trusted certification authorities, or disable the common name verification. See TLSOptions.client and TLSOptions.client_unsafe.

---

void **disconnect_from_stream**() 🔗

Disconnects from host.

Status **get_status**() const 🔗

Returns the status of the connection. See Status for values.

StreamPeer **get_stream**() const 🔗

Returns the underlying StreamPeer connection, used in accept_stream or connect_to_stream.

---

void **poll**() 🔗

Poll the connection to check for incoming bytes. Call this right before StreamPeer.get_available_bytes for it to work properly.

SurfaceTool

Inherits: RefCounted **<** Object

Helper tool to create geometry.

Description

The **SurfaceTool** is used to construct a Mesh by specifying vertex attributes individually. It can be used to construct a Mesh from a script. All properties except indices need to be added before calling add_vertex. For example, to add vertex colors and UVs:

GDScript

```
var st = SurfaceTool.new()
st.begin(Mesh.PRIMITIVE_TRIANGLES)
st.set_color(Color(1, 0, 0))
st.set_uv(Vector2(0, 0))
st.add_vertex(Vector3(0, 0, 0))
```

C#

```
var st = new SurfaceTool();
st.Begin(Mesh.PrimitiveType.Triangles);
st.SetColor(new Color(1, 0, 0));
st.SetUV(new Vector2(0, 0));
st.AddVertex(new Vector3(0, 0, 0));
```

The above **SurfaceTool** now contains one vertex of a triangle which has a UV coordinate and a specified Color. If another vertex were added without calling set_uv or set_color, then the last values would be used.

Vertex attributes must be passed **before** calling add_vertex. Failure to do so will result in an error when committing the vertex information to a mesh.

Additionally, the attributes used before the first vertex is added determine the format of the mesh. For example, if you only add UVs to the first vertex, you cannot add color to any of the subsequent vertices.

See also ArrayMesh, ImmediateMesh and MeshDataTool for procedural geometry generation.

Note: Godot uses clockwise winding order [https://learnopengl.com/Advanced-OpenGL/Face-culling] for front faces of triangle primitive modes.

Tutorials

- Using the SurfaceTool
- 3D Voxel Demo [https://godotengine.org/asset-library/asset/2755]

Methods

void	add_index(index: int)
void	add_triangle_fan(vertices: PackedVector3Array, uvs: PackedVector2Array = PackedVector2Array(), colors: PackedColorArray = PackedColorArray(), uv2s: PackedVector2Array = PackedVector2Array(), normals: PackedVector3Array =

	PackedVector3Array(), tangents: Array[Plane] = [])
void	add_vertex(vertex: Vector3)
void	append_from(existing: Mesh, surface: int, transform: Transform3D)
void	begin(primitive: PrimitiveType)
void	clear()
ArrayMesh	commit(existing: ArrayMesh = null, flags: int = 0)
Array	commit_to_arrays()
void	create_from(existing: Mesh, surface: int)
void	create_from_arrays(arrays: Array, primitive_type: PrimitiveType = 3)
void	create_from_blend_shape(existing: Mesh, surface: int, blend_shape: String)
void	deindex()

PackedInt32Array	generate_lod(nd_threshold: float, target_index_count: int = 3)
void	generate_normals(flip: bool = false)
void	generate_tangents()
AABB	get_aabb() const
CustomFormat	get_custom_format(channel_index: int) const
PrimitiveType	get_primitive_type() const
SkinWeightCount	get_skin_weight_count() const
void	index()
void	optimize_indices_for_cache()
void	set_bones(bones: PackedInt32Array)
void	set_color(color: Color)
void	set_custom(channel_index: int, custom_color: Color)

void	set_custom_format(channel_index: int, format: CustomFormat)
void	set_material(material: Material)
void	set_normal(normal: Vector3)
void	set_skin_weight_count(count: SkinWeightCount)
void	set_smooth_group(index: int)
void	set_tangent(tangent: Plane)
void	set_uv(uv: Vector2)
void	set_uv2(uv2: Vector2)
void	set_weights(weights: PackedFloat32Array)

Enumerations

enum **CustomFormat**: 🔗

CustomFormat **CUSTOM_RGBA8_UNORM** = 0

Limits range of data passed to set_custom to unsigned normalized 0 to 1 stored in 8 bits per channel. See Mesh.ARRAY_CUSTOM_RGBA8_UNORM.

CustomFormat **CUSTOM_RGBA8_SNORM** = `1`

Limits range of data passed to set_custom to signed normalized -1 to 1 stored in 8 bits per channel. See Mesh.ARRAY_CUSTOM_RGBA8_SNORM.

CustomFormat **CUSTOM_RG_HALF** = `2`

Stores data passed to set_custom as half precision floats, and uses only red and green color channels. See Mesh.ARRAY_CUSTOM_RG_HALF.

CustomFormat **CUSTOM_RGBA_HALF** = `3`

Stores data passed to set_custom as half precision floats and uses all color channels. See Mesh.ARRAY_CUSTOM_RGBA_HALF.

CustomFormat **CUSTOM_R_FLOAT** = `4`

Stores data passed to set_custom as full precision floats, and uses only red color channel. See Mesh.ARRAY_CUSTOM_R_FLOAT.

CustomFormat **CUSTOM_RG_FLOAT** = `5`

Stores data passed to set_custom as full precision floats, and uses only red and green color channels. See Mesh.ARRAY_CUSTOM_RG_FLOAT.

CustomFormat **CUSTOM_RGB_FLOAT** = `6`

Stores data passed to set_custom as full precision floats, and uses only red, green and blue color channels. See Mesh.ARRAY_CUSTOM_RGB_FLOAT.

CustomFormat **CUSTOM_RGBA_FLOAT** = `7`

Stores data passed to set_custom as full precision floats, and uses all color channels. See Mesh.ARRAY_CUSTOM_RGBA_FLOAT.

CustomFormat **CUSTOM_MAX** = `8`

Used to indicate a disabled custom channel.

enum **SkinWeightCount**: 🔗

SkinWeightCount **SKIN_4_WEIGHTS** = `0`

Each individual vertex can be influenced by only 4 bone weights.

SkinWeightCount **SKIN_8_WEIGHTS** = `1`

Each individual vertex can be influenced by up to 8 bone weights.

Method Descriptions

void **add_index**(index: int) 🔗

Adds a vertex to index array if you are using indexed vertices. Does not need to be called before adding vertices.

void **add_triangle_fan**(vertices: PackedVector3Array, uvs: PackedVector2Array = PackedVector2Array(), colors: PackedColorArray = PackedColorArray(), uv2s: PackedVector2Array = PackedVector2Array(), normals: PackedVector3Array = PackedVector3Array(), tangents: Array[Plane] = []) 🔗

Inserts a triangle fan made of array data into Mesh being constructed.

Requires the primitive type be set to Mesh.PRIMITIVE_TRIANGLES.

---

void **add_vertex**(vertex: Vector3)

Specifies the position of current vertex. Should be called after specifying other vertex properties (e.g. Color, UV).

---

void **append_from**(existing: Mesh, surface: int, transform: Transform3D)

Append vertices from a given Mesh surface onto the current vertex array with specified Transform3D.

---

void **begin**(primitive: PrimitiveType)

Called before adding any vertices. Takes the primitive type as an argument (e.g. Mesh.PRIMITIVE_TRIANGLES).

---

void **clear**()

Clear all information passed into the surface tool so far.

ArrayMesh **commit**(existing: ArrayMesh = null, flags: int = 0)

Returns a constructed ArrayMesh from current information passed in. If an existing ArrayMesh is passed in as an argument, will add an extra surface to the existing ArrayMesh.

FIXME: Document possible values for `flags`, it changed in 4.0. Likely some combinations of ArrayFormat.

Array **commit_to_arrays**()

Commits the data to the same format used by ArrayMesh.add_surface_from_arrays, ImporterMesh.add_surface, and create_from_arrays. This way you can further process the mesh data using the ArrayMesh or ImporterMesh APIs.

---

void **create_from**(existing: Mesh, surface: int)

Creates a vertex array from an existing Mesh.

---

void **create_from_arrays**(arrays: Array, primitive_type: PrimitiveType = 3)

Creates this SurfaceTool from existing vertex arrays such as returned by commit_to_arrays, Mesh.surface_get_arrays, Mesh.surface_get_blend_shape_arrays, ImporterMesh.get_surface_arrays, and ImporterMesh.get_surface_blend_shape_arrays. `primitive_type` controls the type of mesh data, defaulting to Mesh.PRIMITIVE_TRIANGLES.

---

void **create_from_blend_shape**(existing: Mesh, surface: int, blend_shape: String)

Creates a vertex array from the specified blend shape of an existing Mesh. This can be used to extract a specific pose from a blend shape.

---

void **deindex**()

Removes the index array by expanding the vertex array.

PackedInt32Array **generate_lod**(nd_threshold: float, target_index_count: int = 3)

Deprecated: This method is unused internally, as it does not preserve normals or UVs. Consider using ImporterMesh.generate_lods instead.

Generates an LOD for a given `nd_threshold` in linear units (square root of quadric error metric), using at most `target_index_count` indices.

---

void **generate_normals**(flip: bool = false)

Generates normals from vertices so you do not have to do it manually. If `flip` is `true`, the resulting normals will be inverted. generate_normals should be called *after* generating geometry and *before* committing the mesh using commit or commit_to_arrays. For correct display of normal-mapped surfaces, you will also have to generate tangents using generate_tangents.

Note: generate_normals only works if the primitive type to be set to Mesh.PRIMITIVE_TRIANGLES.

Note: generate_normals takes smooth groups into account. To generate smooth normals, set the smooth group to a value greater than or equal to `0` using set_smooth_group or leave the smooth group at the default of `0`. To generate flat normals, set the smooth group to `-1` using set_smooth_group prior to adding vertices.

---

void **generate_tangents**() 🔗

Generates a tangent vector for each vertex. Requires that each vertex have UVs and normals set already (see generate_normals).

AABB **get_aabb**() const 🔗

Returns the axis-aligned bounding box of the vertex positions.

CustomFormat **get_custom_format**(channel_index: int) const 🔗

Returns the format for custom `channel_index` (currently up to 4). Returns CUSTOM_MAX if this custom channel is unused.

PrimitiveType **get_primitive_type**() const 🔗

Returns the type of mesh geometry, such as Mesh.PRIMITIVE_TRIANGLES.

SkinWeightCount **get_skin_weight_count**() const 🔗

By default, returns SKIN_4_WEIGHTS to indicate only 4 bone influences per vertex are used.

Returns SKIN_8_WEIGHTS if up to 8 influences are used.

Note: This function returns an enum, not the exact number of weights.

void **index**() 🔗

Shrinks the vertex array by creating an index array. This can improve performance by avoiding vertex reuse.

void **optimize_indices_for_cache**()

Optimizes triangle sorting for performance. Requires that get_primitive_type is Mesh.PRIMITIVE_TRIANGLES.

void **set_bones**(bones: PackedInt32Array)

Specifies an array of bones to use for the *next* vertex. `bones` must contain 4 integers.

void **set_color**(color: Color)

Specifies a Color to use for the *next* vertex. If every vertex needs to have this information set and you fail to submit it for the first vertex, this information may not be used at all.

Note: The material must have BaseMaterial3D.vertex_color_use_as_albedo enabled for the vertex color to be visible.

void **set_custom**(channel_index: int, custom_color: Color)

Sets the custom value on this vertex for `channel_index`.

set_custom_format must be called first for this `channel_index`. Formats which are not RGBA will ignore other color channels.

void **set_custom_format**(channel_index: int, format: CustomFormat)

Sets the color format for this custom `channel_index`. Use CUSTOM_MAX to disable.

Must be invoked after begin and should be set before commit or commit_to_arrays.

---

void **set_material**(material: Material)

Sets Material to be used by the Mesh you are constructing.

---

void **set_normal**(normal: Vector3)

Specifies a normal to use for the *next* vertex. If every vertex needs to have this information set and you fail to submit it for the first vertex, this information may not be used at all.

---

void **set_skin_weight_count**(count: SkinWeightCount)

Set to SKIN_8_WEIGHTS to indicate that up to 8 bone influences per vertex may be used.

By default, only 4 bone influences are used (SKIN_4_WEIGHTS)

Note: This function takes an enum, not the exact number of weights.

---

void **set_smooth_group**(index: int)

Specifies the smooth group to use for the *next* vertex. If this is never called, all vertices will have the default smooth group of `0` and will be smoothed with adjacent vertices of the same group. To produce a mesh with flat normals, set the smooth group to `-1`.

Note: This function actually takes a `uint32_t`, so C# users should use `uint32.MaxValue` instead of `-1` to produce a mesh with flat normals.

---

void **set_tangent**(tangent: Plane)

Specifies a tangent to use for the *next* vertex. If every vertex needs to have this information set and you fail to submit it for the first vertex, this information may not be used at all.

---

void **set_uv**(uv: Vector2)

Specifies a set of UV coordinates to use for the *next* vertex. If every vertex needs to have this information set and you fail to submit it for the first vertex, this information may not be used at all.

---

void **set_uv2**(uv2: Vector2)

Specifies an optional second set of UV coordinates to use for the *next* vertex. If every vertex needs to have this information set and you fail to submit it for the first vertex, this information may not be used at all.

---

void **set_weights**(weights: PackedFloat32Array)

Specifies weight values to use for the *next* vertex. `weights` must contain 4 values. If every vertex needs to have this information set and you fail to submit it for the first vertex, this information may not be used at all.

TCPServer

Inherits: RefCounted **<** Object

A TCP server.

Description

A TCP server. Listens to connections on a port and returns a StreamPeerTCP when it gets an incoming connection.

Note: When exporting to Android, make sure to enable the `INTERNET` permission in the Android export preset before exporting the project or using one-click deploy. Otherwise, network communication of any kind will be blocked by Android.

Methods

int	get_local_port() const
bool	is_connection_available() const
bool	is_listening() const
Error	listen(port: int, bind_address: String = "*")
void	stop()

StreamPeerTCP take_connection()

Method Descriptions

int **get_local_port**() const

Returns the local port this server is listening to.

---

bool **is_connection_available**() const

Returns `true` if a connection is available for taking.

---

bool **is_listening**() const

Returns `true` if the server is currently listening for connections.

Error **listen**(port: int, bind_address: String = "*")

Listen on the `port` binding to `bind_address`.

If `bind_address` is set as `"*"` (default), the server will listen on all available addresses (both IPv4 and IPv6).

If `bind_address` is set as `"0.0.0.0"` (for IPv4) or `"::"` (for IPv6), the server will listen on all available addresses matching that IP type.

If `bind_address` is set to any valid address (e.g. `"192.168.1.101"`, `"::1"`, etc.), the server will only listen on the interface with that address (or fail if no interface with the given address exists).

---

void **stop**()

Stops listening.

StreamPeerTCP **take_connection**()

If a connection is available, returns a StreamPeerTCP with the connection.

TextLine

Inherits: RefCounted **<** Object

Holds a line of text.

Description

Abstraction over TextServer for handling a single line of text.

Properties

HorizontalAlignment	alignment	`0`
Direction	direction	`0`
String	ellipsis_char	`"…"`
BitField[JustificationFlag]	flags	`3`
Orientation	orientation	`0`
bool	preserve_control	`false`
bool	preserve_invalid	`true`

OverrunBehavior	text_overrun_behavior	`3`
float	width	`-1.0`

Methods

bool	add_object(key: Variant, size: Vector2, inline_align: InlineAlignment = 5, length: int = 1, baseline: float = 0.0)
bool	add_string(text: String, font: Font, font_size: int, language: String = "", meta: Variant = null)
void	clear()
void	draw(canvas: RID, pos: Vector2, color: Color = Color(1, 1, 1, 1)) const
void	draw_outline(canvas: RID, pos: Vector2, outline_size: int = 1, color: Color = Color(1, 1, 1, 1)) const
float	get_line_ascent() const
float	get_line_descent() const
float	get_line_underline_position() const

float	get_line_underline_thickness() const
float	get_line_width() const
Rect2	get_object_rect(key: Variant) const
Array	get_objects() const
RID	get_rid() const
Vector2	get_size() const
int	hit_test(coords: float) const
bool	resize_object(key: Variant, size: Vector2, inline_align: InlineAlignment = 5, baseline: float = 0.0)
void	set_bidi_override(override: Array)
void	tab_align(tab_stops: PackedFloat32Array)

Property Descriptions

HorizontalAlignment **alignment** = 0 🔗

- void **set_horizontal_alignment**(value: HorizontalAlignment)
- HorizontalAlignment **get_horizontal_alignment**()

Sets text alignment within the line as if the line was horizontal.

Direction **direction** = `0` 🔗

- void **set_direction**(value: Direction)
- Direction **get_direction**()

Text writing direction.

String **ellipsis_char** = `"…"` 🔗

- void **set_ellipsis_char**(value: String)
- String **get_ellipsis_char**()

Ellipsis character used for text clipping.

BitField[JustificationFlag] **flags** = `3` 🔗

- void **set_flags**(value: BitField[JustificationFlag])
- BitField[JustificationFlag] **get_flags**()

Line alignment rules. For more info see TextServer.

Orientation **orientation** = `0` 🔗

- void **set_orientation**(value: Orientation)
- Orientation **get_orientation**()

Text orientation.

---

bool **preserve_control** = `false`

- void **set_preserve_control**(value: bool)
- bool **get_preserve_control**()

If set to `true` text will display control characters.

---

bool **preserve_invalid** = `true`

- void **set_preserve_invalid**(value: bool)
- bool **get_preserve_invalid**()

If set to `true` text will display invalid characters.

OverrunBehavior **text_overrun_behavior** = `3`

- void **set_text_overrun_behavior**(value: OverrunBehavior)
- OverrunBehavior **get_text_overrun_behavior**()

Sets the clipping behavior when the text exceeds the text line's set width. See OverrunBehavior for a description of all modes.

---

float **width** = `-1.0`

- void **set_width**(value: float)
- float **get_width**()

Text line width.

Method Descriptions

bool **add_object**(key: Variant, size: Vector2, inline_align: InlineAlignment = 5, length: int = 1, baseline: float = 0.0) 🔗

Adds inline object to the text buffer, `key` must be unique. In the text, object is represented as `length` object replacement characters.

bool **add_string**(text: String, font: Font, font_size: int, language: String = "", meta: Variant = null) 🔗

Adds text span and font to draw it.

void **clear**() 🔗

Clears text line (removes text and inline objects).

void **draw**(canvas: RID, pos: Vector2, color: Color = Color(1, 1, 1, 1)) const 🔗

Draw text into a canvas item at a given position, with `color`. `pos` specifies the top left corner of the bounding box.

void **draw_outline**(canvas: RID, pos: Vector2, outline_size: int = 1, color: Color = Color(1, 1, 1, 1)) const 🔗

Draw text into a canvas item at a given position, with `color`. `pos` specifies the top left corner of the bounding box.

float **get_line_ascent**() const 🔗

Returns the text ascent (number of pixels above the baseline for horizontal layout or to the left of baseline for vertical).

---

float **get_line_descent**() const

Returns the text descent (number of pixels below the baseline for horizontal layout or to the right of baseline for vertical).

---

float **get_line_underline_position**() const

Returns pixel offset of the underline below the baseline.

---

float **get_line_underline_thickness**() const

Returns thickness of the underline.

---

float **get_line_width**() const

Returns width (for horizontal layout) or height (for vertical) of the text.

Rect2 **get_object_rect**(key: Variant) const

Returns bounding rectangle of the inline object.

Array **get_objects**() const

Returns array of inline objects.

RID **get_rid**() const

Returns TextServer buffer RID.

Vector2 **get_size**() const

Returns size of the bounding box of the text.

int **hit_test**(coords: float) const

Returns caret character offset at the specified pixel offset at the baseline. This function always returns a valid position.

bool **resize_object**(key: Variant, size: Vector2, inline_align: InlineAlignment = 5, baseline: float = 0.0)

Sets new size and alignment of embedded object.

void **set_bidi_override**(override: Array)

Overrides BiDi for the structured text.

Override ranges should cover full source text without overlaps. BiDi algorithm will be used on each range separately.

void **tab_align**(tab_stops: PackedFloat32Array)

Aligns text to the given tab-stops.

TextParagraph

Inherits: RefCounted **<** Object

Holds a paragraph of text.

Description

Abstraction over TextServer for handling a single paragraph of text.

Properties

HorizontalAlignment	alignment	`0`
BitField[LineBreakFlag]	break_flags	`3`
String	custom_punctuation	`""`
Direction	direction	`0`
String	ellipsis_char	`"…"`
BitField[JustificationFlag]	justification_flags	`163`
int	max_lines_visible	`-1`

Orientation	orientation	`0`
bool	preserve_control	`false`
bool	preserve_invalid	`true`
OverrunBehavior	text_overrun_behavior	`0`
float	width	`-1.0`

Methods

bool	add_object(key: Variant, size: Vector2, inline_align: InlineAlignment = 5, length: int = 1, baseline: float = 0.0)
bool	add_string(text: String, font: Font, font_size: int, language: String = "", meta: Variant = null)
void	clear()
void	clear_dropcap()
void	draw(canvas: RID, pos: Vector2, color: Color = Color(1, 1, 1, 1), dc_color: Color = Color(1, 1, 1, 1)) const

void	draw_dropcap(canvas: RID, pos: Vector2, color: Color = Color(1, 1, 1, 1)) const
void	draw_dropcap_outline(canvas: RID, pos: Vector2, outline_size: int = 1, color: Color = Color(1, 1, 1, 1)) const
void	draw_line(canvas: RID, pos: Vector2, line: int, color: Color = Color(1, 1, 1, 1)) const
void	draw_line_outline(canvas: RID, pos: Vector2, line: int, outline_size: int = 1, color: Color = Color(1, 1, 1, 1)) const
void	draw_outline(canvas: RID, pos: Vector2, outline_size: int = 1, color: Color = Color(1, 1, 1, 1), dc_color: Color = Color(1, 1, 1, 1)) const
int	get_dropcap_lines() const
RID	get_dropcap_rid() const
Vector2	get_dropcap_size() const
float	get_line_ascent(line: int) const

int	get_line_count() const
float	get_line_descent(line: int) const
Rect2	get_line_object_rect(line: int, key: Variant) const
Array	get_line_objects(line: int) const
Vector2i	get_line_range(line: int) const
RID	get_line_rid(line: int) const
Vector2	get_line_size(line: int) const
float	get_line_underline_position(line: int) const
float	get_line_underline_thickness(line: int) const
float	get_line_width(line: int) const
Vector2	get_non_wrapped_size() const
RID	get_rid() const
Vector2	get_size() const

int	hit_test(coords: Vector2) const
bool	resize_object(key: Variant, size: Vector2, inline_align: InlineAlignment = 5, baseline: float = 0.0)
void	set_bidi_override(override: Array)
bool	set_dropcap(text: String, font: Font, font_size: int, dropcap_margins: Rect2 = Rect2(0, 0, 0, 0), language: String = "")
void	tab_align(tab_stops: PackedFloat32Array)

Property Descriptions

HorizontalAlignment **alignment** = 0 🔗

- void **set_alignment**(value: HorizontalAlignment)
- HorizontalAlignment **get_alignment**()

Paragraph horizontal alignment.

BitField[LineBreakFlag] **break_flags** = 3 🔗

- void **set_break_flags**(value: BitField[LineBreakFlag])
- BitField[LineBreakFlag] **get_break_flags**()

Line breaking rules. For more info see TextServer.

String **custom_punctuation** = `""`

- void **set_custom_punctuation**(value: String)
- String **get_custom_punctuation**()

Custom punctuation character list, used for word breaking. If set to empty string, server defaults are used.

Direction **direction** = `0`

- void **set_direction**(value: Direction)
- Direction **get_direction**()

Text writing direction.

String **ellipsis_char** = `"…"`

- void **set_ellipsis_char**(value: String)
- String **get_ellipsis_char**()

Ellipsis character used for text clipping.

BitField[JustificationFlag] **justification_flags** = `163`

- void **set_justification_flags**(value: BitField[JustificationFlag])
- BitField[JustificationFlag] **get_justification_flags**()

Line fill alignment rules. For more info see JustificationFlag.

---

int **max_lines_visible** = `-1`

- void **set_max_lines_visible**(value: int)

- int **get_max_lines_visible**()

Limits the lines of text shown.

Orientation **orientation** = `0`

- void **set_orientation**(value: Orientation)
- Orientation **get_orientation**()

Text orientation.

---

bool **preserve_control** = `false`

- void **set_preserve_control**(value: bool)
- bool **get_preserve_control**()

If set to `true` text will display control characters.

---

bool **preserve_invalid** = `true`

- void **set_preserve_invalid**(value: bool)
- bool **get_preserve_invalid**()

If set to `true` text will display invalid characters.

OverrunBehavior **text_overrun_behavior** = `0`

- void **set_text_overrun_behavior**(value: OverrunBehavior)
- OverrunBehavior **get_text_overrun_behavior**()

Sets the clipping behavior when the text exceeds the paragraph's set width. See OverrunBehavior for a description of all modes.

---

float **width** = `-1.0`

- void **set_width**(value: float)
- float **get_width**()

Paragraph width.

Method Descriptions

bool **add_object**(key: Variant, size: Vector2, inline_align: InlineAlignment = 5, length: int = 1, baseline: float = 0.0)

Adds inline object to the text buffer, `key` must be unique. In the text, object is represented as `length` object replacement characters.

---

bool **add_string**(text: String, font: Font, font_size: int, language: String = "", meta: Variant = null)

Adds text span and font to draw it.

---

void **clear**()

Clears text paragraph (removes text and inline objects).

---

void **clear_dropcap**()

Removes dropcap.

---

void **draw**(canvas: RID, pos: Vector2, color: Color = Color(1, 1, 1, 1), dc_color: Color = Color(1, 1, 1, 1)) const

Draw all lines of the text and drop cap into a canvas item at a given position, with `color`. `pos` specifies the top left corner of the bounding box.

---

void **draw_dropcap**(canvas: RID, pos: Vector2, color: Color = Color(1, 1, 1, 1)) const

Draw drop cap into a canvas item at a given position, with `color`. `pos` specifies the top left corner of the bounding box.

---

void **draw_dropcap_outline**(canvas: RID, pos: Vector2, outline_size: int = 1, color: Color = Color(1, 1, 1, 1)) const

Draw drop cap outline into a canvas item at a given position, with `color`. `pos` specifies the top left corner of the bounding box.

---

void **draw_line**(canvas: RID, pos: Vector2, line: int, color: Color = Color(1, 1, 1, 1)) const

Draw single line of text into a canvas item at a given position, with `color`. `pos` specifies the top left corner of the bounding box.

---

void **draw_line_outline**(canvas: RID, pos: Vector2, line: int, outline_size: int = 1, color: Color = Color(1, 1, 1, 1)) const

Draw outline of the single line of text into a canvas item at a given position, with `color`. `pos` specifies the top left corner of the bounding box.

---

void **draw_outline**(canvas: RID, pos: Vector2, outline_size: int = 1, color: Color = Color(1, 1, 1, 1), dc_color: Color = Color(1, 1, 1, 1)) const 🔗

Draw outlines of all lines of the text and drop cap into a canvas item at a given position, with `color`. `pos` specifies the top left corner of the bounding box.

---

int **get_dropcap_lines**() const 🔗

Returns number of lines used by dropcap.

RID **get_dropcap_rid**() const 🔗

Returns drop cap text buffer RID.

Vector2 **get_dropcap_size**() const 🔗

Returns drop cap bounding box size.

---

float **get_line_ascent**(line: int) const 🔗

Returns the text line ascent (number of pixels above the baseline for horizontal layout or to the left of baseline for vertical).

---

int **get_line_count**() const 🔗

Returns number of lines in the paragraph.

---

float **get_line_descent**(line: int) const 🔗

Returns the text line descent (number of pixels below the baseline for horizontal layout or to the right of baseline for vertical).

Rect2 **get_line_object_rect**(line: int, key: Variant) const 🔗

Returns bounding rectangle of the inline object.

Array **get_line_objects**(line: int) const 🔗

Returns array of inline objects in the line.

Vector2i **get_line_range**(line: int) const 🔗

Returns character range of the line.

RID **get_line_rid**(line: int) const 🔗

Returns TextServer line buffer RID.

Vector2 **get_line_size**(line: int) const 🔗

Returns size of the bounding box of the line of text. Returned size is rounded up.

---

float **get_line_underline_position**(line: int) const 🔗

Returns pixel offset of the underline below the baseline.

---

float **get_line_underline_thickness**(line: int) const 🔗

Returns thickness of the underline.

---

float **get_line_width**(line: int) const 🔗

Returns width (for horizontal layout) or height (for vertical) of the line of text.

Vector2 **get_non_wrapped_size**() const 🔗

Returns the size of the bounding box of the paragraph, without line breaks.

RID **get_rid**() const 🔗

Returns TextServer full string buffer RID.

Vector2 **get_size**() const 🔗

Returns the size of the bounding box of the paragraph.

int **hit_test**(coords: Vector2) const 🔗

Returns caret character offset at the specified coordinates. This function always returns a valid position.

bool **resize_object**(key: Variant, size: Vector2, inline_align: InlineAlignment = 5, baseline: float = 0.0) 🔗

Sets new size and alignment of embedded object.

void **set_bidi_override**(override: Array) 🔗

Overrides BiDi for the structured text.

Override ranges should cover full source text without overlaps. BiDi algorithm will be used on each range separately.

---

bool **set_dropcap**(text: String, font: Font, font_size: int, dropcap_margins: Rect2 = Rect2(0, 0, 0, 0), language: String = "")

Sets drop cap, overrides previously set drop cap. Drop cap (dropped capital) is a decorative element at the beginning of a paragraph that is larger than the rest of the text.

---

void **tab_align**(tab_stops: PackedFloat32Array)

Aligns paragraph to the given tab-stops.

TextServer

Inherits: RefCounted < Object

Inherited By: TextServerExtension

A server interface for font management and text rendering.

Description

TextServer is the API backend for managing fonts and rendering text.

Note: This is a low-level API, consider using TextLine, TextParagraph, and Font classes instead.

This is an abstract class, so to get the currently active **TextServer** instance, use the following code:

GDScript

```gdscript
var ts = TextServerManager.get_primary_interface()
```

C#

```csharp
var ts = TextServerManager.GetPrimaryInterface();
```

Methods

RID	create_font()
RID	create_font_linked_variation(font_rid: RID)
RID	create_shaped_text(direction: Direction = 0, orientation: Orientation = 0)

void	draw_hex_code_box(canvas: RID, size: int, pos: Vector2, index: int, color: Color) const
void	font_clear_glyphs(font_rid: RID, size: Vector2i)
void	font_clear_kerning_map(font_rid: RID, size: int)
void	font_clear_size_cache(font_rid: RID)
void	font_clear_textures(font_rid: RID, size: Vector2i)
void	font_draw_glyph(font_rid: RID, canvas: RID, size: int, pos: Vector2, index: int, color: Color = Color(1, 1, 1, 1)) const
void	font_draw_glyph_outline(font_rid: RID, canvas: RID, size: int, outline_size: int, pos: Vector2, index: int, color: Color = Color(1, 1, 1, 1)) const
FontAntialiasing	font_get_antialiasing(font_rid: RID) const
float	font_get_ascent(font_rid: RID, size: int) const
float	font_get_baseline_offset(font_rid: RID) const
int	font_get_char_from_glyph_index(font_rid: RID, size: int, glyph_index: int) const
float	font_get_descent(font_rid: RID, size: int) const

bool	font_get_disable_embedded_bitmaps(font_rid: RID) const
float	font_get_embolden(font_rid: RID) const
int	font_get_face_count(font_rid: RID) const
int	font_get_face_index(font_rid: RID) const
int	font_get_fixed_size(font_rid: RID) const
FixedSizeScaleMode	font_get_fixed_size_scale_mode(font_rid: RID) const
bool	font_get_generate_mipmaps(font_rid: RID) const
float	font_get_global_oversampling() const
Vector2	font_get_glyph_advance(font_rid: RID, size: int, glyph: int) const
Dictionary	font_get_glyph_contours(font: RID, size: int, index: int) const
int	font_get_glyph_index(font_rid: RID, size: int, char: int, variation_selector: int) const
PackedInt32Array	font_get_glyph_list(font_rid: RID, size: Vector2i) const
Vector2	font_get_glyph_offset(font_rid: RID, size: Vector2i, glyph: int) const

Vector2	font_get_glyph_size(font_rid: RID, size: Vector2i, glyph: int) const
int	font_get_glyph_texture_idx(font_rid: RID, size: Vector2i, glyph: int) const
RID	font_get_glyph_texture_rid(font_rid: RID, size: Vector2i, glyph: int) const
Vector2	font_get_glyph_texture_size(font_rid: RID, size: Vector2i, glyph: int) const
Rect2	font_get_glyph_uv_rect(font_rid: RID, size: Vector2i, glyph: int) const
Hinting	font_get_hinting(font_rid: RID) const
Vector2	font_get_kerning(font_rid: RID, size: int, glyph_pair: Vector2i) const
Array[Vector2i]	font_get_kerning_list(font_rid: RID, size: int) const
bool	font_get_language_support_override(font_rid: RID, language: String)
PackedStringArray	font_get_language_support_overrides(font_rid: RID)
int	font_get_msdf_pixel_range(font_rid: RID) const
int	font_get_msdf_size(font_rid: RID) const

String	font_get_name(font_rid: RID) const
Dictionary	font_get_opentype_feature_overrides(font_rid: RID) const
Dictionary	font_get_ot_name_strings(font_rid: RID) const
float	font_get_oversampling(font_rid: RID) const
float	font_get_scale(font_rid: RID, size: int) const
bool	font_get_script_support_override(font_rid: RID, script: String)
PackedStringArray	font_get_script_support_overrides(font_rid: RID)
Array[Vector2i]	font_get_size_cache_list(font_rid: RID) const
int	font_get_spacing(font_rid: RID, spacing: SpacingType) const
int	font_get_stretch(font_rid: RID) const
BitField[FontStyle]	font_get_style(font_rid: RID) const
String	font_get_style_name(font_rid: RID) const
SubpixelPositioning	font_get_subpixel_positioning(font_rid: RID) const

String	font_get_supported_chars(font_rid: RID) const
int	font_get_texture_count(font_rid: RID, size: Vector2i) const
Image	font_get_texture_image(font_rid: RID, size: Vector2i, texture_index: int) const
PackedInt32Array	font_get_texture_offsets(font_rid: RID, size: Vector2i, texture_index: int) const
Transform2D	font_get_transform(font_rid: RID) const
float	font_get_underline_position(font_rid: RID, size: int) const
float	font_get_underline_thickness(font_rid: RID, size: int) const
Dictionary	font_get_variation_coordinates(font_rid: RID) const
int	font_get_weight(font_rid: RID) const
bool	font_has_char(font_rid: RID, char: int) const
bool	font_is_allow_system_fallback(font_rid: RID) const
bool	font_is_force_autohinter(font_rid: RID) const

bool	font_is_language_supported(font_rid: RID, language: String) const
bool	font_is_multichannel_signed_distance_field(font_rid: RID) const
bool	font_is_script_supported(font_rid: RID, script: String) const
void	font_remove_glyph(font_rid: RID, size: Vector2i, glyph: int)
void	font_remove_kerning(font_rid: RID, size: int, glyph_pair: Vector2i)
void	font_remove_language_support_override(font_rid: RID, language: String)
void	font_remove_script_support_override(font_rid: RID, script: String)
void	font_remove_size_cache(font_rid: RID, size: Vector2i)
void	font_remove_texture(font_rid: RID, size: Vector2i, texture_index: int)
void	font_render_glyph(font_rid: RID, size: Vector2i, index: int)
void	font_render_range(font_rid: RID, size: Vector2i, start: int, end: int)

void	font_set_allow_system_fallback(font_rid: RID, allow_system_fallback: bool)
void	font_set_antialiasing(font_rid: RID, antialiasing: FontAntialiasing)
void	font_set_ascent(font_rid: RID, size: int, ascent: float)
void	font_set_baseline_offset(font_rid: RID, baseline_offset: float)
void	font_set_data(font_rid: RID, data: PackedByteArray)
void	font_set_descent(font_rid: RID, size: int, descent: float)
void	font_set_disable_embedded_bitmaps(font_rid: RID, disable_embedded_bitmaps: bool)
void	font_set_embolden(font_rid: RID, strength: float)
void	font_set_face_index(font_rid: RID, face_index: int)
void	font_set_fixed_size(font_rid: RID, fixed_size: int)
void	font_set_fixed_size_scale_mode(font_rid: RID, fixed_size_scale_mode: FixedSizeScaleMode)
void	font_set_force_autohinter(font_rid: RID, force_autohinter: bool)

void	font_set_generate_mipmaps(font_rid: RID, generate_mipmaps: bool)
void	font_set_global_oversampling(oversampling: float)
void	font_set_glyph_advance(font_rid: RID, size: int, glyph: int, advance: Vector2)
void	font_set_glyph_offset(font_rid: RID, size: Vector2i, glyph: int, offset: Vector2)
void	font_set_glyph_size(font_rid: RID, size: Vector2i, glyph: int, gl_size: Vector2)
void	font_set_glyph_texture_idx(font_rid: RID, size: Vector2i, glyph: int, texture_idx: int)
void	font_set_glyph_uv_rect(font_rid: RID, size: Vector2i, glyph: int, uv_rect: Rect2)
void	font_set_hinting(font_rid: RID, hinting: Hinting)
void	font_set_kerning(font_rid: RID, size: int, glyph_pair: Vector2i, kerning: Vector2)
void	font_set_language_support_override(font_rid: RID, language: String, supported: bool)
void	font_set_msdf_pixel_range(font_rid: RID, msdf_pixel_range: int)

void	font_set_msdf_size(font_rid: RID, msdf_size: int)
void	font_set_multichannel_signed_distance_field(font_rid: RID, msdf: bool)
void	font_set_name(font_rid: RID, name: String)
void	font_set_opentype_feature_overrides(font_rid: RID, overrides: Dictionary)
void	font_set_oversampling(font_rid: RID, oversampling: float)
void	font_set_scale(font_rid: RID, size: int, scale: float)
void	font_set_script_support_override(font_rid: RID, script: String, supported: bool)
void	font_set_spacing(font_rid: RID, spacing: SpacingType, value: int)
void	font_set_stretch(font_rid: RID, weight: int)
void	font_set_style(font_rid: RID, style: BitField[FontStyle])
void	font_set_style_name(font_rid: RID, name: String)
void	font_set_subpixel_positioning(font_rid: RID, subpixel_positioning: SubpixelPositioning)

void	font_set_texture_image(font_rid: RID, size: Vector2i, texture_index: int, image: Image)
void	font_set_texture_offsets(font_rid: RID, size: Vector2i, texture_index: int, offset: PackedInt32Array)
void	font_set_transform(font_rid: RID, transform: Transform2D)
void	font_set_underline_position(font_rid: RID, size: int, underline_position: float)
void	font_set_underline_thickness(font_rid: RID, size: int, underline_thickness: float)
void	font_set_variation_coordinates(font_rid: RID, variation_coordinates: Dictionary)
void	font_set_weight(font_rid: RID, weight: int)
Dictionary	font_supported_feature_list(font_rid: RID) const
Dictionary	font_supported_variation_list(font_rid: RID) const
String	format_number(number: String, language: String = "") const
void	free_rid(rid: RID)

int	get_features() const
Vector2	get_hex_code_box_size(size: int, index: int) const
String	get_name() const
String	get_support_data_filename() const
String	get_support_data_info() const
bool	has(rid: RID)
bool	has_feature(feature: Feature) const
int	is_confusable(string: String, dict: PackedStringArray) const
bool	is_locale_right_to_left(locale: String) const
bool	is_valid_identifier(string: String) const
bool	is_valid_letter(unicode: int) const
bool	load_support_data(filename: String)
int	name_to_tag(name: String) const
String	parse_number(number: String, language: String = "") const

Array[Vector3i]	parse_structured_text(parser_type: StructuredTextParser, args: Array, text: String) const
String	percent_sign(language: String = "") const
bool	save_support_data(filename: String) const
int	shaped_get_span_count(shaped: RID) const
Variant	shaped_get_span_meta(shaped: RID, index: int) const
void	shaped_set_span_update_font(shaped: RID, index: int, fonts: Array[RID], size: int, opentype_features: Dictionary = {})
bool	shaped_text_add_object(shaped: RID, key: Variant, size: Vector2, inline_align: InlineAlignment = 5, length: int = 1, baseline: float = 0.0)
bool	shaped_text_add_string(shaped: RID, text: String, fonts: Array[RID], size: int, opentype_features: Dictionary = {}, language: String = "", meta: Variant = null)
void	shaped_text_clear(rid: RID)
int	shaped_text_closest_character_pos(shaped: RID, pos: int) const
void	shaped_text_draw(shaped: RID, canvas: RID, pos: Vector2, clip_l: float = -1, clip_r: float = -1, color: Color

	= Color(1, 1, 1, 1)) const
void	shaped_text_draw_outline(shaped: RID, canvas: RID, pos: Vector2, clip_l: float = -1, clip_r: float = -1, outline_size: int = 1, color: Color = Color(1, 1, 1, 1)) const
float	shaped_text_fit_to_width(shaped: RID, width: float, justification_flags: BitField[JustificationFlag] = 3)
float	shaped_text_get_ascent(shaped: RID) const
Dictionary	shaped_text_get_carets(shaped: RID, position: int) const
PackedInt32Array	shaped_text_get_character_breaks(shaped: RID) const
int	shaped_text_get_custom_ellipsis(shaped: RID) const
String	shaped_text_get_custom_punctuation(shaped: RID) const
float	shaped_text_get_descent(shaped: RID) const
Direction	shaped_text_get_direction(shaped: RID) const
Direction	shaped_text_get_dominant_direction_in_range(shaped: RID, start: int, end: int) const
int	shaped_text_get_ellipsis_glyph_count(shaped: RID) const

Array[Dictionary]	shaped_text_get_ellipsis_glyphs(shaped: RID) const
int	shaped_text_get_ellipsis_pos(shaped: RID) const
int	shaped_text_get_glyph_count(shaped: RID) const
Array[Dictionary]	shaped_text_get_glyphs(shaped: RID) const
Vector2	shaped_text_get_grapheme_bounds(shaped: RID, pos: int) const
Direction	shaped_text_get_inferred_direction(shaped: RID) const
PackedInt32Array	shaped_text_get_line_breaks(shaped: RID, width: float, start: int = 0, break_flags: BitField[LineBreakFlag] = 3) const
PackedInt32Array	shaped_text_get_line_breaks_adv(shaped: RID, width: PackedFloat32Array, start: int = 0, once: bool = true, break_flags: BitField[LineBreakFlag] = 3) const
int	shaped_text_get_object_glyph(shaped: RID, key: Variant) const
Vector2i	shaped_text_get_object_range(shaped: RID, key: Variant) const
Rect2	shaped_text_get_object_rect(shaped: RID, key: Variant) const

Array	shaped_text_get_objects(shaped: RID) const
Orientation	shaped_text_get_orientation(shaped: RID) const
RID	shaped_text_get_parent(shaped: RID) const
bool	shaped_text_get_preserve_control(shaped: RID) const
bool	shaped_text_get_preserve_invalid(shaped: RID) const
Vector2i	shaped_text_get_range(shaped: RID) const
PackedVector2Array	shaped_text_get_selection(shaped: RID, start: int, end: int) const
Vector2	shaped_text_get_size(shaped: RID) const
int	shaped_text_get_spacing(shaped: RID, spacing: SpacingType) const
int	shaped_text_get_trim_pos(shaped: RID) const
float	shaped_text_get_underline_position(shaped: RID) const
float	shaped_text_get_underline_thickness(shaped: RID) const
float	shaped_text_get_width(shaped: RID) const

PackedInt32Array	shaped_text_get_word_breaks(shaped: RID, grapheme_flags: BitField[GraphemeFlag] = 264, skip_grapheme_flags: BitField[GraphemeFlag] = 4) const
bool	shaped_text_has_visible_chars(shaped: RID) const
int	shaped_text_hit_test_grapheme(shaped: RID, coords: float) const
int	shaped_text_hit_test_position(shaped: RID, coords: float) const
bool	shaped_text_is_ready(shaped: RID) const
int	shaped_text_next_character_pos(shaped: RID, pos: int) const
int	shaped_text_next_grapheme_pos(shaped: RID, pos: int) const
void	shaped_text_overrun_trim_to_width(shaped: RID, width: float = 0, overrun_trim_flags: BitField[TextOverrunFlag] = 0)
int	shaped_text_prev_character_pos(shaped: RID, pos: int) const
int	shaped_text_prev_grapheme_pos(shaped: RID, pos: int) const

bool	shaped_text_resize_object(shaped: RID, key: Variant, size: Vector2, inline_align: InlineAlignment = 5, baseline: float = 0.0)
void	shaped_text_set_bidi_override(shaped: RID, override: Array)
void	shaped_text_set_custom_ellipsis(shaped: RID, char: int)
void	shaped_text_set_custom_punctuation(shaped: RID, punct: String)
void	shaped_text_set_direction(shaped: RID, direction: Direction = 0)
void	shaped_text_set_orientation(shaped: RID, orientation: Orientation = 0)
void	shaped_text_set_preserve_control(shaped: RID, enabled: bool)
void	shaped_text_set_preserve_invalid(shaped: RID, enabled: bool)
void	shaped_text_set_spacing(shaped: RID, spacing: SpacingType, value: int)
bool	shaped_text_shape(shaped: RID)
Array[Dictionary]	shaped_text_sort_logical(shaped: RID)

RID	shaped_text_substr(shaped: RID, start: int, length: int) const
float	shaped_text_tab_align(shaped: RID, tab_stops: PackedFloat32Array)
bool	spoof_check(string: String) const
PackedInt32Array	string_get_character_breaks(string: String, language: String = "") const
PackedInt32Array	string_get_word_breaks(string: String, language: String = "", chars_per_line: int = 0) const
String	string_to_lower(string: String, language: String = "") const
String	string_to_title(string: String, language: String = "") const
String	string_to_upper(string: String, language: String = "") const
String	strip_diacritics(string: String) const
String	tag_to_name(tag: int) const

Enumerations

enum **FontAntialiasing**:

FontAntialiasing **FONT_ANTIALIASING_NONE** = `0`

Font glyphs are rasterized as 1-bit bitmaps.

FontAntialiasing **FONT_ANTIALIASING_GRAY** = `1`

Font glyphs are rasterized as 8-bit grayscale anti-aliased bitmaps.

FontAntialiasing **FONT_ANTIALIASING_LCD** = `2`

Font glyphs are rasterized for LCD screens.

LCD subpixel layout is determined by the value of `gui/theme/lcd_subpixel_layout` project settings.

LCD subpixel anti-aliasing mode is suitable only for rendering horizontal, unscaled text in 2D.

enum **FontLCDSubpixelLayout**:

FontLCDSubpixelLayout **FONT_LCD_SUBPIXEL_LAYOUT_NONE** = `0`

Unknown or unsupported subpixel layout, LCD subpixel antialiasing is disabled.

FontLCDSubpixelLayout **FONT_LCD_SUBPIXEL_LAYOUT_HRGB** = `1`

Horizontal RGB subpixel layout.

FontLCDSubpixelLayout **FONT_LCD_SUBPIXEL_LAYOUT_HBGR** = `2`

Horizontal BGR subpixel layout.

FontLCDSubpixelLayout **FONT_LCD_SUBPIXEL_LAYOUT_VRGB** = `3`

Vertical RGB subpixel layout.

FontLCDSubpixelLayout **FONT_LCD_SUBPIXEL_LAYOUT_VBGR** = `4`

Vertical BGR subpixel layout.

FontLCDSubpixelLayout **FONT_LCD_SUBPIXEL_LAYOUT_MAX** = `5`

Represents the size of the FontLCDSubpixelLayout enum.

---

enum **Direction**: 🔗

Direction **DIRECTION_AUTO** = `0`

Text direction is determined based on contents and current locale.

Direction **DIRECTION_LTR** = `1`

Text is written from left to right.

Direction **DIRECTION_RTL** = `2`

Text is written from right to left.

Direction **DIRECTION_INHERITED** = `3`

Text writing direction is the same as base string writing direction. Used for BiDi override only.

---

enum **Orientation**: 🔗

Orientation **ORIENTATION_HORIZONTAL** = `0`

Text is written horizontally.

Orientation **ORIENTATION_VERTICAL** = `1`

Left to right text is written vertically from top to bottom.

Right to left text is written vertically from bottom to top.

---

flags **JustificationFlag**: 🔗

JustificationFlag **JUSTIFICATION_NONE** = `0`

Do not justify text.

JustificationFlag **JUSTIFICATION_KASHIDA** = `1`

Justify text by adding and removing kashidas.

JustificationFlag **JUSTIFICATION_WORD_BOUND** = `2`

Justify text by changing width of the spaces between the words.

JustificationFlag **JUSTIFICATION_TRIM_EDGE_SPACES** = `4`

Remove trailing and leading spaces from the justified text.

JustificationFlag **JUSTIFICATION_AFTER_LAST_TAB** = `8`

Only apply justification to the part of the text after the last tab.

JustificationFlag **JUSTIFICATION_CONSTRAIN_ELLIPSIS** = `16`

Apply justification to the trimmed line with ellipsis.

JustificationFlag **JUSTIFICATION_SKIP_LAST_LINE** = `32`

Do not apply justification to the last line of the paragraph.

JustificationFlag
JUSTIFICATION_SKIP_LAST_LINE_WITH_VISIBLE_CHARS = `64`

Do not apply justification to the last line of the paragraph with visible characters (takes precedence over JUSTIFICATION_SKIP_LAST_LINE).

JustificationFlag **JUSTIFICATION_DO_NOT_SKIP_SINGLE_LINE** = `128`

Always apply justification to the paragraphs with a single line (JUSTIFICATION_SKIP_LAST_LINE and JUSTIFICATION_SKIP_LAST_LINE_WITH_VISIBLE_CHARS are ignored).

---

enum **AutowrapMode**: 🔗

AutowrapMode **AUTOWRAP_OFF** = `0`

Autowrap is disabled.

AutowrapMode **AUTOWRAP_ARBITRARY** = `1`

Wraps the text inside the node's bounding rectangle by allowing to break lines at arbitrary positions, which is useful when very limited space is available.

AutowrapMode **AUTOWRAP_WORD** = `2`

Wraps the text inside the node's bounding rectangle by soft-breaking between words.

AutowrapMode **AUTOWRAP_WORD_SMART** = `3`

Behaves similarly to AUTOWRAP_WORD, but force-breaks a word if that single word does not fit in one line.

---

flags **LineBreakFlag**: 🔗

LineBreakFlag **BREAK_NONE** = `0`

Do not break the line.

LineBreakFlag **BREAK_MANDATORY** = `1`

Break the line at the line mandatory break characters (e.g. `"\n"`).

LineBreakFlag **BREAK_WORD_BOUND** = `2`

Break the line between the words.

LineBreakFlag **BREAK_GRAPHEME_BOUND** = `4`

Break the line between any unconnected graphemes.

LineBreakFlag **BREAK_ADAPTIVE** = `8`

Should be used only in conjunction with BREAK_WORD_BOUND, break the line between any unconnected graphemes, if it's impossible to break it between the words.

LineBreakFlag **BREAK_TRIM_EDGE_SPACES** = `16`

Remove edge spaces from the broken line segments.

LineBreakFlag **BREAK_TRIM_INDENT** = `32`

Subtract first line indentation width from all lines after the first one.

---

enum **VisibleCharactersBehavior**: 🔗

VisibleCharactersBehavior **VC_CHARS_BEFORE_SHAPING** = `0`

Trims text before the shaping. e.g, increasing Label.visible_characters or RichTextLabel.visible_characters value is visually identical to typing the text.

VisibleCharactersBehavior **VC_CHARS_AFTER_SHAPING** = `1`

Displays glyphs that are mapped to the first Label.visible_characters or RichTextLabel.visible_characters characters from the beginning of the text.

VisibleCharactersBehavior **VC_GLYPHS_AUTO** = `2`

Displays Label.visible_ratio or RichTextLabel.visible_ratio glyphs, starting from the left or from the right, depending on Control.layout_direction value.

VisibleCharactersBehavior **VC_GLYPHS_LTR** = `3`

Displays Label.visible_ratio or RichTextLabel.visible_ratio glyphs, starting from the left.

VisibleCharactersBehavior **VC_GLYPHS_RTL** = `4`

Displays Label.visible_ratio or RichTextLabel.visible_ratio glyphs, starting from the right.

---

enum **OverrunBehavior**: 🔗

OverrunBehavior **OVERRUN_NO_TRIMMING** = `0`

No text trimming is performed.

OverrunBehavior **OVERRUN_TRIM_CHAR** = `1`

Trims the text per character.

OverrunBehavior **OVERRUN_TRIM_WORD** = `2`

Trims the text per word.

OverrunBehavior **OVERRUN_TRIM_ELLIPSIS** = `3`

Trims the text per character and adds an ellipsis to indicate that parts are hidden.

OverrunBehavior **OVERRUN_TRIM_WORD_ELLIPSIS** = `4`

Trims the text per word and adds an ellipsis to indicate that parts are hidden.

---

flags **TextOverrunFlag**: 🔗

TextOverrunFlag **OVERRUN_NO_TRIM** = `0`

No trimming is performed.

TextOverrunFlag **OVERRUN_TRIM** = `1`

Trims the text when it exceeds the given width.

TextOverrunFlag **OVERRUN_TRIM_WORD_ONLY** = `2`

Trims the text per word instead of per grapheme.

TextOverrunFlag **OVERRUN_ADD_ELLIPSIS** = `4`

Determines whether an ellipsis should be added at the end of the text.

TextOverrunFlag **OVERRUN_ENFORCE_ELLIPSIS** = `8`

Determines whether the ellipsis at the end of the text is enforced and may not be hidden.

TextOverrunFlag **OVERRUN_JUSTIFICATION_AWARE** = `16`

Accounts for the text being justified before attempting to trim it (see JustificationFlag).

---

flags **GraphemeFlag**: 🔗

GraphemeFlag **GRAPHEME_IS_VALID** = `1`

Grapheme is supported by the font, and can be drawn.

GraphemeFlag **GRAPHEME_IS_RTL** = `2`

Grapheme is part of right-to-left or bottom-to-top run.

GraphemeFlag **GRAPHEME_IS_VIRTUAL** = `4`

Grapheme is not part of source text, it was added by justification process.

GraphemeFlag **GRAPHEME_IS_SPACE** = `8`

Grapheme is whitespace.

GraphemeFlag **GRAPHEME_IS_BREAK_HARD** = `16`

Grapheme is mandatory break point (e.g. `"\n"`).

GraphemeFlag **GRAPHEME_IS_BREAK_SOFT** = `32`

Grapheme is optional break point (e.g. space).

GraphemeFlag **GRAPHEME_IS_TAB** = `64`

Grapheme is the tabulation character.

GraphemeFlag **GRAPHEME_IS_ELONGATION** = `128`

Grapheme is kashida.

GraphemeFlag **GRAPHEME_IS_PUNCTUATION** = `256`

Grapheme is punctuation character.

GraphemeFlag **GRAPHEME_IS_UNDERSCORE** = `512`

Grapheme is underscore character.

GraphemeFlag **GRAPHEME_IS_CONNECTED** = `1024`

Grapheme is connected to the previous grapheme. Breaking line before this grapheme is not safe.

GraphemeFlag **GRAPHEME_IS_SAFE_TO_INSERT_TATWEEL** = `2048`

It is safe to insert a U+0640 before this grapheme for elongation.

GraphemeFlag **GRAPHEME_IS_EMBEDDED_OBJECT** = `4096`

Grapheme is an object replacement character for the embedded object.

GraphemeFlag **GRAPHEME_IS_SOFT_HYPHEN** = `8192`

Grapheme is a soft hyphen.

enum **Hinting**: 🔗

Hinting **HINTING_NONE** = `0`

Disables font hinting (smoother but less crisp).

Hinting **HINTING_LIGHT** = `1`

Use the light font hinting mode.

Hinting **HINTING_NORMAL** = `2`

Use the default font hinting mode (crisper but less smooth).

Note: This hinting mode changes both horizontal and vertical glyph metrics. If applied to monospace font, some glyphs might have different width.

enum **SubpixelPositioning**: 🔗

SubpixelPositioning **SUBPIXEL_POSITIONING_DISABLED** = `0`

Glyph horizontal position is rounded to the whole pixel size, each glyph is rasterized once.

SubpixelPositioning **SUBPIXEL_POSITIONING_AUTO** = `1`

Glyph horizontal position is rounded based on font size.

- To one quarter of the pixel size if font size is smaller or equal to SUBPIXEL_POSITIONING_ONE_QUARTER_MAX_SIZE.
- To one half of the pixel size if font size is smaller or equal to SUBPIXEL_POSITIONING_ONE_HALF_MAX_SIZE.
- To the whole pixel size for larger fonts.

SubpixelPositioning **SUBPIXEL_POSITIONING_ONE_HALF** = `2`

Glyph horizontal position is rounded to one half of the pixel size, each glyph is rasterized up to two times.

SubpixelPositioning **SUBPIXEL_POSITIONING_ONE_QUARTER** = `3`

Glyph horizontal position is rounded to one quarter of the pixel size, each glyph is rasterized up to four times.

SubpixelPositioning **SUBPIXEL_POSITIONING_ONE_HALF_MAX_SIZE** = `20`

Maximum font size which will use one half of the pixel subpixel positioning in SUBPIXEL_POSITIONING_AUTO mode.

SubpixelPositioning **SUBPIXEL_POSITIONING_ONE_QUARTER_MAX_SIZE** = `16`

Maximum font size which will use one quarter of the pixel subpixel positioning in SUBPIXEL_POSITIONING_AUTO mode.

---

enum **Feature**: 🔗

Feature **FEATURE_SIMPLE_LAYOUT** = `1`

TextServer supports simple text layouts.

Feature **FEATURE_BIDI_LAYOUT** = `2`

TextServer supports bidirectional text layouts.

Feature **FEATURE_VERTICAL_LAYOUT** = `4`

TextServer supports vertical layouts.

Feature **FEATURE_SHAPING** = `8`

TextServer supports complex text shaping.

Feature **FEATURE_KASHIDA_JUSTIFICATION** = `16`

TextServer supports justification using kashidas.

Feature **FEATURE_BREAK_ITERATORS** = `32`

TextServer supports complex line/word breaking rules (e.g. dictionary based).

Feature **FEATURE_FONT_BITMAP** = `64`

TextServer supports loading bitmap fonts.

Feature **FEATURE_FONT_DYNAMIC** = `128`

TextServer supports loading dynamic (TrueType, OpeType, etc.) fonts.

Feature **FEATURE_FONT_MSDF** = `256`

TextServer supports multichannel signed distance field dynamic font rendering.

Feature **FEATURE_FONT_SYSTEM** = `512`

TextServer supports loading system fonts.

Feature **FEATURE_FONT_VARIABLE** = `1024`

TextServer supports variable fonts.

Feature **FEATURE_CONTEXT_SENSITIVE_CASE_CONVERSION** = `2048`

TextServer supports locale dependent and context sensitive case conversion.

Feature **FEATURE_USE_SUPPORT_DATA** = `4096`

TextServer require external data file for some features, see load_support_data.

Feature **FEATURE_UNICODE_IDENTIFIERS** = `8192`

TextServer supports UAX #31 identifier validation, see is_valid_identifier.

Feature **FEATURE_UNICODE_SECURITY** = `16384`

TextServer supports Unicode Technical Report #36 [https://unicode.org/reports/tr36/] and Unicode Technical Standard #39 [https://unicode.org/reports/tr39/] based spoof detection features.

enum **ContourPointTag**: 🔗

ContourPointTag **CONTOUR_CURVE_TAG_ON** = `1`

Contour point is on the curve.

ContourPointTag **CONTOUR_CURVE_TAG_OFF_CONIC** = `0`

Contour point isn't on the curve, but serves as a control point for a conic (quadratic) Bézier arc.

ContourPointTag **CONTOUR_CURVE_TAG_OFF_CUBIC** = `2`

Contour point isn't on the curve, but serves as a control point for a cubic Bézier arc.

enum **SpacingType**: 🔗

SpacingType **SPACING_GLYPH** = `0`

Spacing for each glyph.

SpacingType **SPACING_SPACE** = `1`

Spacing for the space character.

SpacingType **SPACING_TOP** = `2`

Spacing at the top of the line.

SpacingType **SPACING_BOTTOM** = `3`

Spacing at the bottom of the line.

SpacingType **SPACING_MAX** = `4`

Represents the size of the SpacingType enum.

---

flags **FontStyle**: 🔗

FontStyle **FONT_BOLD** = `1`

Font is bold.

FontStyle **FONT_ITALIC** = `2`

Font is italic or oblique.

FontStyle **FONT_FIXED_WIDTH** = `4`

Font have fixed-width characters.

---

enum **StructuredTextParser**: 🔗

StructuredTextParser **STRUCTURED_TEXT_DEFAULT** = `0`

Use default Unicode BiDi algorithm.

StructuredTextParser **STRUCTURED_TEXT_URI** = `1`

BiDi override for URI.

StructuredTextParser **STRUCTURED_TEXT_FILE** = `2`

BiDi override for file path.

StructuredTextParser **STRUCTURED_TEXT_EMAIL** = `3`

BiDi override for email.

StructuredTextParser **STRUCTURED_TEXT_LIST** = `4`

BiDi override for lists. Structured text options: list separator String.

StructuredTextParser **STRUCTURED_TEXT_GDSCRIPT** = `5`

BiDi override for GDScript.

StructuredTextParser **STRUCTURED_TEXT_CUSTOM** = `6`

User defined structured text BiDi override function.

enum **FixedSizeScaleMode**:

FixedSizeScaleMode **FIXED_SIZE_SCALE_DISABLE** = `0`

Bitmap font is not scaled.

FixedSizeScaleMode **FIXED_SIZE_SCALE_INTEGER_ONLY** = `1`

Bitmap font is scaled to the closest integer multiple of the font's fixed size. This is the recommended option for pixel art fonts.

FixedSizeScaleMode **FIXED_SIZE_SCALE_ENABLED** = `2`

Bitmap font is scaled to an arbitrary (fractional) size. This is the recommended option for non-pixel art fonts.

Method Descriptions

RID **create_font**()

Creates a new, empty font cache entry resource. To free the resulting resource, use the free_rid method.

RID **create_font_linked_variation**(font_rid: RID)

Creates a new variation existing font which is reusing the same glyph cache and font data. To free the resulting resource, use the free_rid method.

RID **create_shaped_text**(direction: Direction = 0, orientation: Orientation = 0)

Creates a new buffer for complex text layout, with the given `direction` and `orientation`. To free the resulting buffer, use free_rid method.

Note: Direction is ignored if server does not support FEATURE_BIDI_LAYOUT feature (supported by TextServerAdvanced).

Note: Orientation is ignored if server does not support FEATURE_VERTICAL_LAYOUT feature (supported by TextServerAdvanced).

---

void **draw_hex_code_box**(canvas: RID, size: int, pos: Vector2, index: int, color: Color) const

Draws box displaying character hexadecimal code. Used for replacing missing characters.

---

void **font_clear_glyphs**(font_rid: RID, size: Vector2i)

Removes all rendered glyph information from the cache entry.

Note: This function will not remove textures associated with the glyphs, use font_remove_texture to remove them manually.

---

void **font_clear_kerning_map**(font_rid: RID, size: int)

Removes all kerning overrides.

---

void **font_clear_size_cache**(font_rid: RID)

Removes all font sizes from the cache entry.

void **font_clear_textures**(font_rid: RID, size: Vector2i)

Removes all textures from font cache entry.

Note: This function will not remove glyphs associated with the texture, use font_remove_glyph to remove them manually.

void **font_draw_glyph**(font_rid: RID, canvas: RID, size: int, pos: Vector2, index: int, color: Color = Color(1, 1, 1, 1)) const

Draws single glyph into a canvas item at the position, using `font_rid` at the size `size`.

Note: Glyph index is specific to the font, use glyphs indices returned by shaped_text_get_glyphs or font_get_glyph_index.

Note: If there are pending glyphs to render, calling this function might trigger the texture cache update.

void **font_draw_glyph_outline**(font_rid: RID, canvas: RID, size: int, outline_size: int, pos: Vector2, index: int, color: Color = Color(1, 1, 1, 1)) const

Draws single glyph outline of size `outline_size` into a canvas item at the position, using `font_rid` at the size `size`.

Note: Glyph index is specific to the font, use glyphs indices returned by shaped_text_get_glyphs or font_get_glyph_index.

Note: If there are pending glyphs to render, calling this function might trigger the texture cache update.

FontAntialiasing **font_get_antialiasing**(font_rid: RID) const

Returns font anti-aliasing mode.

float **font_get_ascent**(font_rid: RID, size: int) const 🔗

Returns the font ascent (number of pixels above the baseline).

---

float **font_get_baseline_offset**(font_rid: RID) const 🔗

Returns extra baseline offset (as a fraction of font height).

---

int **font_get_char_from_glyph_index**(font_rid: RID, size: int, glyph_index: int) const 🔗

Returns character code associated with `glyph_index`, or `0` if `glyph_index` is invalid. See font_get_glyph_index.

---

float **font_get_descent**(font_rid: RID, size: int) const 🔗

Returns the font descent (number of pixels below the baseline).

---

bool **font_get_disable_embedded_bitmaps**(font_rid: RID) const 🔗

Returns whether the font's embedded bitmap loading is disabled.

---

float **font_get_embolden**(font_rid: RID) const 🔗

Returns font embolden strength.

---

int **font_get_face_count**(font_rid: RID) const 🔗

Returns number of faces in the TrueType / OpenType collection.

---

int **font_get_face_index**(font_rid: RID) const 🔗

Returns an active face index in the TrueType / OpenType collection.

int **font_get_fixed_size**(font_rid: RID) const

Returns bitmap font fixed size.

FixedSizeScaleMode **font_get_fixed_size_scale_mode**(font_rid: RID) const

Returns bitmap font scaling mode.

---

bool **font_get_generate_mipmaps**(font_rid: RID) const

Returns `true` if font texture mipmap generation is enabled.

---

float **font_get_global_oversampling**() const

Returns the font oversampling factor, shared by all fonts in the TextServer.

Vector2 **font_get_glyph_advance**(font_rid: RID, size: int, glyph: int) const

Returns glyph advance (offset of the next glyph).

Note: Advance for glyphs outlines is the same as the base glyph advance and is not saved.

Dictionary **font_get_glyph_contours**(font: RID, size: int, index: int) const

Returns outline contours of the glyph as a Dictionary with the following contents:

`points` - PackedVector3Array, containing outline points. `x` and `y` are point coordinates. `z` is the type of the point, using the ContourPointTag values.

`contours` - PackedInt32Array, containing indices the end points of each contour.

`orientation` - bool, contour orientation. If `true`, clockwise contours must be filled.

---

int **font_get_glyph_index**(font_rid: RID, size: int, char: int, variation_selector: int) const

Returns the glyph index of a `char`, optionally modified by the `variation_selector`. See font_get_char_from_glyph_index.

PackedInt32Array **font_get_glyph_list**(font_rid: RID, size: Vector2i) const

Returns list of rendered glyphs in the cache entry.

Vector2 **font_get_glyph_offset**(font_rid: RID, size: Vector2i, glyph: int) const

Returns glyph offset from the baseline.

Vector2 **font_get_glyph_size**(font_rid: RID, size: Vector2i, glyph: int) const

Returns size of the glyph.

---

int **font_get_glyph_texture_idx**(font_rid: RID, size: Vector2i, glyph: int) const

Returns index of the cache texture containing the glyph.

RID **font_get_glyph_texture_rid**(font_rid: RID, size: Vector2i, glyph: int) const

Returns resource ID of the cache texture containing the glyph.

Note: If there are pending glyphs to render, calling this function might trigger the texture cache update.

Vector2 **font_get_glyph_texture_size**(font_rid: RID, size: Vector2i, glyph: int) const 🔗

Returns size of the cache texture containing the glyph.

Note: If there are pending glyphs to render, calling this function might trigger the texture cache update.

Rect2 **font_get_glyph_uv_rect**(font_rid: RID, size: Vector2i, glyph: int) const 🔗

Returns rectangle in the cache texture containing the glyph.

Hinting **font_get_hinting**(font_rid: RID) const 🔗

Returns the font hinting mode. Used by dynamic fonts only.

Vector2 **font_get_kerning**(font_rid: RID, size: int, glyph_pair: Vector2i) const 🔗

Returns kerning for the pair of glyphs.

Array[Vector2i] **font_get_kerning_list**(font_rid: RID, size: int) const 🔗

Returns list of the kerning overrides.

---

bool **font_get_language_support_override**(font_rid: RID, language: String) 🔗

Returns `true` if support override is enabled for the `language`.

PackedStringArray **font_get_language_support_overrides**(font_rid: RID) 🔗

Returns list of language support overrides.

int **font_get_msdf_pixel_range**(font_rid: RID) const 🔗

Returns the width of the range around the shape between the minimum and maximum representable signed distance.

int **font_get_msdf_size**(font_rid: RID) const 🔗

Returns source font size used to generate MSDF textures.

String **font_get_name**(font_rid: RID) const 🔗

Returns font family name.

Dictionary **font_get_opentype_feature_overrides**(font_rid: RID) const 🔗

Returns font OpenType feature set override.

Dictionary **font_get_ot_name_strings**(font_rid: RID) const 🔗

Returns Dictionary with OpenType font name strings (localized font names, version, description, license information, sample text, etc.).

float **font_get_oversampling**(font_rid: RID) const 🔗

Returns font oversampling factor, if set to `0.0` global oversampling factor is used instead. Used by dynamic fonts only.

float **font_get_scale**(font_rid: RID, size: int) const 🔗

Returns scaling factor of the color bitmap font.

---

bool **font_get_script_support_override**(font_rid: RID, script: String)

Returns `true` if support override is enabled for the `script`.

PackedStringArray **font_get_script_support_overrides**(font_rid: RID)

Returns list of script support overrides.

Array[Vector2i] **font_get_size_cache_list**(font_rid: RID) const

Returns list of the font sizes in the cache. Each size is Vector2i with font size and outline size.

---

int **font_get_spacing**(font_rid: RID, spacing: SpacingType) const

Returns the spacing for `spacing` (see SpacingType) in pixels (not relative to the font size).

---

int **font_get_stretch**(font_rid: RID) const

Returns font stretch amount, compared to a normal width. A percentage value between `50%` and `200%`.

BitField[FontStyle] **font_get_style**(font_rid: RID) const

Returns font style flags, see FontStyle.

String **font_get_style_name**(font_rid: RID) const

Returns font style name.

SubpixelPositioning **font_get_subpixel_positioning**(font_rid: RID) const 🔗

Returns font subpixel glyph positioning mode.

String **font_get_supported_chars**(font_rid: RID) const 🔗

Returns a string containing all the characters available in the font.

---

int **font_get_texture_count**(font_rid: RID, size: Vector2i) const 🔗

Returns number of textures used by font cache entry.

Image **font_get_texture_image**(font_rid: RID, size: Vector2i, texture_index: int) const 🔗

Returns font cache texture image data.

PackedInt32Array **font_get_texture_offsets**(font_rid: RID, size: Vector2i, texture_index: int) const 🔗

Returns array containing glyph packing data.

Transform2D **font_get_transform**(font_rid: RID) const 🔗

Returns 2D transform applied to the font outlines.

---

float **font_get_underline_position**(font_rid: RID, size: int) const 🔗

Returns pixel offset of the underline below the baseline.

---

float **font_get_underline_thickness**(font_rid: RID, size: int) const 🔗

Returns thickness of the underline in pixels.

Dictionary **font_get_variation_coordinates**(font_rid: RID) const

Returns variation coordinates for the specified font cache entry. See font_supported_variation_list for more info.

---

int **font_get_weight**(font_rid: RID) const

Returns weight (boldness) of the font. A value in the `100...999` range, normal font weight is `400`, bold font weight is `700`.

---

bool **font_has_char**(font_rid: RID, char: int) const

Returns `true` if a Unicode `char` is available in the font.

---

bool **font_is_allow_system_fallback**(font_rid: RID) const

Returns `true` if system fonts can be automatically used as fallbacks.

---

bool **font_is_force_autohinter**(font_rid: RID) const

Returns `true` if auto-hinting is supported and preferred over font built-in hinting. Used by dynamic fonts only.

---

bool **font_is_language_supported**(font_rid: RID, language: String) const

Returns `true`, if font supports given language (ISO 639 [https://en.wikipedia.org/wiki/ISO_639-1] code).

---

bool **font_is_multichannel_signed_distance_field**(font_rid: RID) const

Returns `true` if glyphs of all sizes are rendered using single multichannel signed distance field generated from the dynamic font vector data.

bool **font_is_script_supported**(font_rid: RID, script: String) const

Returns `true`, if font supports given script (ISO 15924 code).

void **font_remove_glyph**(font_rid: RID, size: Vector2i, glyph: int)

Removes specified rendered glyph information from the cache entry.

Note: This function will not remove textures associated with the glyphs, use font_remove_texture to remove them manually.

void **font_remove_kerning**(font_rid: RID, size: int, glyph_pair: Vector2i)

Removes kerning override for the pair of glyphs.

void **font_remove_language_support_override**(font_rid: RID, language: String)

Remove language support override.

void **font_remove_script_support_override**(font_rid: RID, script: String)

Removes script support override.

void **font_remove_size_cache**(font_rid: RID, size: Vector2i)

Removes specified font size from the cache entry.

void **font_remove_texture**(font_rid: RID, size: Vector2i, texture_index: int)

Removes specified texture from the cache entry.

Note: This function will not remove glyphs associated with the texture, remove them manually, using font_remove_glyph.

---

void **font_render_glyph**(font_rid: RID, size: Vector2i, index: int)

Renders specified glyph to the font cache texture.

---

void **font_render_range**(font_rid: RID, size: Vector2i, start: int, end: int)

Renders the range of characters to the font cache texture.

---

void **font_set_allow_system_fallback**(font_rid: RID, allow_system_fallback: bool)

If set to `true`, system fonts can be automatically used as fallbacks.

---

void **font_set_antialiasing**(font_rid: RID, antialiasing: FontAntialiasing)

Sets font anti-aliasing mode.

---

void **font_set_ascent**(font_rid: RID, size: int, ascent: float)

Sets the font ascent (number of pixels above the baseline).

---

void **font_set_baseline_offset**(font_rid: RID, baseline_offset: float)

Sets extra baseline offset (as a fraction of font height).

---

void **font_set_data**(font_rid: RID, data: PackedByteArray)

Sets font source data, e.g contents of the dynamic font source file.

---

void **font_set_descent**(font_rid: RID, size: int, descent: float) 🔗

Sets the font descent (number of pixels below the baseline).

void **font_set_disable_embedded_bitmaps**(font_rid: RID, disable_embedded_bitmaps: bool) 🔗

If set to `true`, embedded font bitmap loading is disabled (bitmap-only and color fonts ignore this property).

void **font_set_embolden**(font_rid: RID, strength: float) 🔗

Sets font embolden strength. If `strength` is not equal to zero, emboldens the font outlines. Negative values reduce the outline thickness.

void **font_set_face_index**(font_rid: RID, face_index: int) 🔗

Sets an active face index in the TrueType / OpenType collection.

void **font_set_fixed_size**(font_rid: RID, fixed_size: int) 🔗

Sets bitmap font fixed size. If set to value greater than zero, same cache entry will be used for all font sizes.

void **font_set_fixed_size_scale_mode**(font_rid: RID, fixed_size_scale_mode: FixedSizeScaleMode) 🔗

Sets bitmap font scaling mode. This property is used only if `fixed_size` is greater than zero.

void **font_set_force_autohinter**(font_rid: RID, force_autohinter: bool) 🔗

If set to `true` auto-hinting is preferred over font built-in hinting.

void **font_set_generate_mipmaps**(font_rid: RID, generate_mipmaps: bool) 🔗

If set to `true` font texture mipmap generation is enabled.

---

void **font_set_global_oversampling**(oversampling: float) 🔗

Sets oversampling factor, shared by all font in the TextServer.

Note: This value can be automatically changed by display server.

---

void **font_set_glyph_advance**(font_rid: RID, size: int, glyph: int, advance: Vector2) 🔗

Sets glyph advance (offset of the next glyph).

Note: Advance for glyphs outlines is the same as the base glyph advance and is not saved.

---

void **font_set_glyph_offset**(font_rid: RID, size: Vector2i, glyph: int, offset: Vector2) 🔗

Sets glyph offset from the baseline.

---

void **font_set_glyph_size**(font_rid: RID, size: Vector2i, glyph: int, gl_size: Vector2) 🔗

Sets size of the glyph.

---

void **font_set_glyph_texture_idx**(font_rid: RID, size: Vector2i, glyph: int, texture_idx: int) 🔗

Sets index of the cache texture containing the glyph.

---

void **font_set_glyph_uv_rect**(font_rid: RID, size: Vector2i, glyph: int, uv_rect: Rect2) 🔗

Sets rectangle in the cache texture containing the glyph.

void **font_set_hinting**(font_rid: RID, hinting: Hinting) 🔗

Sets font hinting mode. Used by dynamic fonts only.

void **font_set_kerning**(font_rid: RID, size: int, glyph_pair: Vector2i, kerning: Vector2) 🔗

Sets kerning for the pair of glyphs.

void **font_set_language_support_override**(font_rid: RID, language: String, supported: bool) 🔗

Adds override for font_is_language_supported.

void **font_set_msdf_pixel_range**(font_rid: RID, msdf_pixel_range: int) 🔗

Sets the width of the range around the shape between the minimum and maximum representable signed distance.

void **font_set_msdf_size**(font_rid: RID, msdf_size: int) 🔗

Sets source font size used to generate MSDF textures.

void **font_set_multichannel_signed_distance_field**(font_rid: RID, msdf: bool) 🔗

If set to `true`, glyphs of all sizes are rendered using single multichannel signed distance field generated from the dynamic font vector data. MSDF rendering allows displaying the font at any scaling factor without blurriness, and without incurring a CPU cost when the font size changes (since the font no longer needs to be rasterized on the CPU). As a downside, font hinting is not available with MSDF. The lack of font hinting may result in less crisp and less readable fonts at small sizes.

Note: MSDF font rendering does not render glyphs with overlapping shapes correctly. Overlapping shapes are not valid per the OpenType standard, but are still commonly found in many font files, especially those converted by Google Fonts. To avoid issues with overlapping glyphs, consider downloading the font file directly from the type foundry instead of relying on Google Fonts.

---

void **font_set_name**(font_rid: RID, name: String) 🔗

Sets the font family name.

---

void **font_set_opentype_feature_overrides**(font_rid: RID, overrides: Dictionary) 🔗

Sets font OpenType feature set override.

---

void **font_set_oversampling**(font_rid: RID, oversampling: float) 🔗

Sets font oversampling factor, if set to `0.0` global oversampling factor is used instead. Used by dynamic fonts only.

---

void **font_set_scale**(font_rid: RID, size: int, scale: float) 🔗

Sets scaling factor of the color bitmap font.

---

void **font_set_script_support_override**(font_rid: RID, script: String, supported: bool) 🔗

Adds override for font_is_script_supported.

---

void **font_set_spacing**(font_rid: RID, spacing: SpacingType, value: int) 🔗

Sets the spacing for `spacing` (see SpacingType) to `value` in pixels (not relative to the font size).

---

void **font_set_stretch**(font_rid: RID, weight: int)

Sets font stretch amount, compared to a normal width. A percentage value between `50%` and `200%`.

Note: This value is used for font matching only and will not affect font rendering. Use font_set_face_index, font_set_variation_coordinates, or font_set_transform instead.

---

void **font_set_style**(font_rid: RID, style: BitField[FontStyle])

Sets the font style flags, see FontStyle.

Note: This value is used for font matching only and will not affect font rendering. Use font_set_face_index, font_set_variation_coordinates, font_set_embolden, or font_set_transform instead.

---

void **font_set_style_name**(font_rid: RID, name: String)

Sets the font style name.

---

void **font_set_subpixel_positioning**(font_rid: RID, subpixel_positioning: SubpixelPositioning)

Sets font subpixel glyph positioning mode.

---

void **font_set_texture_image**(font_rid: RID, size: Vector2i, texture_index: int, image: Image)

Sets font cache texture image data.

---

void **font_set_texture_offsets**(font_rid: RID, size: Vector2i, texture_index: int, offset: PackedInt32Array) 🔗

Sets array containing glyph packing data.

void **font_set_transform**(font_rid: RID, transform: Transform2D) 🔗

Sets 2D transform, applied to the font outlines, can be used for slanting, flipping, and rotating glyphs.

For example, to simulate italic typeface by slanting, apply the following transform `Transform2D(1.0, slant, 0.0, 1.0, 0.0, 0.0)`.

void **font_set_underline_position**(font_rid: RID, size: int, underline_position: float) 🔗

Sets pixel offset of the underline below the baseline.

void **font_set_underline_thickness**(font_rid: RID, size: int, underline_thickness: float) 🔗

Sets thickness of the underline in pixels.

void **font_set_variation_coordinates**(font_rid: RID, variation_coordinates: Dictionary) 🔗

Sets variation coordinates for the specified font cache entry. See font_supported_variation_list for more info.

void **font_set_weight**(font_rid: RID, weight: int) 🔗

Sets weight (boldness) of the font. A value in the `100...999` range, normal font weight is `400`, bold font weight is `700`.

Note: This value is used for font matching only and will not affect font rendering. Use font_set_face_index, font_set_variation_coordinates, or font_set_embolden instead.

Dictionary **font_supported_feature_list**(font_rid: RID) const

Returns the dictionary of the supported OpenType features.

Dictionary **font_supported_variation_list**(font_rid: RID) const

Returns the dictionary of the supported OpenType variation coordinates.

String **format_number**(number: String, language: String = "") const

Converts a number from the Western Arabic (0..9) to the numeral systems used in `language`.

If `language` is omitted, the active locale will be used.

---

void **free_rid**(rid: RID)

Frees an object created by this **TextServer**.

---

int **get_features**() const

Returns text server features, see Feature.

Vector2 **get_hex_code_box_size**(size: int, index: int) const

Returns size of the replacement character (box with character hexadecimal code that is drawn in place of invalid characters).

String **get_name**() const

Returns the name of the server interface.

String **get_support_data_filename**() const 🔗

Returns default TextServer database (e.g. ICU break iterators and dictionaries) filename.

String **get_support_data_info**() const 🔗

Returns TextServer database (e.g. ICU break iterators and dictionaries) description.

---

bool **has**(rid: RID) 🔗

Returns `true` if `rid` is valid resource owned by this text server.

---

bool **has_feature**(feature: Feature) const 🔗

Returns `true` if the server supports a feature.

---

int **is_confusable**(string: String, dict: PackedStringArray) const 🔗

Returns index of the first string in `dict` which is visually confusable with the `string`, or `-1` if none is found.

Note: This method doesn't detect invisible characters, for spoof detection use it in combination with spoof_check.

Note: Always returns `-1` if the server does not support the FEATURE_UNICODE_SECURITY feature.

---

bool **is_locale_right_to_left**(locale: String) const 🔗

Returns `true` if locale is right-to-left.

---

bool **is_valid_identifier**(string: String) const 🔗

Returns `true` if `string` is a valid identifier.

If the text server supports the FEATURE_UNICODE_IDENTIFIERS feature, a valid identifier must:

- Conform to normalization form C.
- Begin with a Unicode character of class XID_Start or `"_"`.
- May contain Unicode characters of class XID_Continue in the other positions.
- Use UAX #31 recommended scripts only (mixed scripts are allowed).

If the FEATURE_UNICODE_IDENTIFIERS feature is not supported, a valid identifier must:

- Begin with a Unicode character of class XID_Start or `"_"`.
- May contain Unicode characters of class XID_Continue in the other positions.

---

bool **is_valid_letter**(unicode: int) const 🔗

Returns `true` if the given code point is a valid letter, i.e. it belongs to the Unicode category "L".

---

bool **load_support_data**(filename: String) 🔗

Loads optional TextServer database (e.g. ICU break iterators and dictionaries).

Note: This function should be called before any other TextServer functions used, otherwise it won't have any effect.

---

int **name_to_tag**(name: String) const 🔗

Converts readable feature, variation, script, or language name to OpenType tag.

String **parse_number**(number: String, language: String = "") const 🔗

Converts `number` from the numeral systems used in `language` to Western Arabic (0..9).

Array[Vector3i] **parse_structured_text**(parser_type: StructuredTextParser, args: Array, text: String) const 🔗

Default implementation of the BiDi algorithm override function. See StructuredTextParser for more info.

String **percent_sign**(language: String = "") const 🔗

Returns percent sign used in the `language`.

---

bool **save_support_data**(filename: String) const 🔗

Saves optional TextServer database (e.g. ICU break iterators and dictionaries) to the file.

Note: This function is used by during project export, to include TextServer database.

---

int **shaped_get_span_count**(shaped: RID) const 🔗

Returns number of text spans added using shaped_text_add_string or shaped_text_add_object.

Variant **shaped_get_span_meta**(shaped: RID, index: int) const 🔗

Returns text span metadata.

---

void **shaped_set_span_update_font**(shaped: RID, index: int, fonts: Array[RID], size: int, opentype_features: Dictionary = {}) 🔗

Changes text span font, font size, and OpenType features, without changing the text.

---

bool **shaped_text_add_object**(shaped: RID, key: Variant, size: Vector2, inline_align: InlineAlignment = 5, length: int = 1, baseline: float = 0.0) 🔗

Adds inline object to the text buffer, `key` must be unique. In the text, object is represented as `length` object replacement characters.

---

bool **shaped_text_add_string**(shaped: RID, text: String, fonts: Array[RID], size: int, opentype_features: Dictionary = {}, language: String = "", meta: Variant = null) 🔗

Adds text span and font to draw it to the text buffer.

---

void **shaped_text_clear**(rid: RID) 🔗

Clears text buffer (removes text and inline objects).

---

int **shaped_text_closest_character_pos**(shaped: RID, pos: int) const 🔗

Returns composite character position closest to the `pos`.

---

void **shaped_text_draw**(shaped: RID, canvas: RID, pos: Vector2, clip_l: float = -1, clip_r: float = -1, color: Color = Color(1, 1, 1, 1)) const 🔗

Draw shaped text into a canvas item at a given position, with `color`. `pos` specifies the leftmost point of the baseline (for horizontal layout) or topmost point of the baseline (for vertical layout).

---

void **shaped_text_draw_outline**(shaped: RID, canvas: RID, pos: Vector2, clip_l: float = -1, clip_r: float = -1, outline_size: int = 1, color: Color = Color(1, 1, 1, 1)) const 🔗

Draw the outline of the shaped text into a canvas item at a given position, with `color`. `pos` specifies the leftmost point of the baseline (for horizontal layout) or topmost point of the baseline (for vertical layout).

---

float **shaped_text_fit_to_width**(shaped: RID, width: float, justification_flags: BitField[JustificationFlag] = 3)

Adjusts text width to fit to specified width, returns new text width.

---

float **shaped_text_get_ascent**(shaped: RID) const

Returns the text ascent (number of pixels above the baseline for horizontal layout or to the left of baseline for vertical).

Note: Overall ascent can be higher than font ascent, if some glyphs are displaced from the baseline.

Dictionary **shaped_text_get_carets**(shaped: RID, position: int) const

Returns shapes of the carets corresponding to the character offset `position` in the text. Returned caret shape is 1 pixel wide rectangle.

PackedInt32Array **shaped_text_get_character_breaks**(shaped: RID) const

Returns array of the composite character boundaries.

---

int **shaped_text_get_custom_ellipsis**(shaped: RID) const

Returns ellipsis character used for text clipping.

String **shaped_text_get_custom_punctuation**(shaped: RID) const

Returns custom punctuation character list, used for word breaking. If set to empty string, server defaults are used.

---

float **shaped_text_get_descent**(shaped: RID) const

Returns the text descent (number of pixels below the baseline for horizontal layout or to the right of baseline for vertical).

Note: Overall descent can be higher than font descent, if some glyphs are displaced from the baseline.

Direction **shaped_text_get_direction**(shaped: RID) const

Returns direction of the text.

Direction **shaped_text_get_dominant_direction_in_range**(shaped: RID, start: int, end: int) const

Returns dominant direction of in the range of text.

---

int **shaped_text_get_ellipsis_glyph_count**(shaped: RID) const

Returns number of glyphs in the ellipsis.

Array[Dictionary] **shaped_text_get_ellipsis_glyphs**(shaped: RID) const

Returns array of the glyphs in the ellipsis.

---

int **shaped_text_get_ellipsis_pos**(shaped: RID) const

Returns position of the ellipsis.

---

int **shaped_text_get_glyph_count**(shaped: RID) const

Returns number of glyphs in the buffer.

Array[Dictionary] **shaped_text_get_glyphs**(shaped: RID) const

Returns an array of glyphs in the visual order.

Vector2 **shaped_text_get_grapheme_bounds**(shaped: RID, pos: int) const

Returns composite character's bounds as offsets from the start of the line.

Direction **shaped_text_get_inferred_direction**(shaped: RID) const

Returns direction of the text, inferred by the BiDi algorithm.

PackedInt32Array **shaped_text_get_line_breaks**(shaped: RID, width: float, start: int = 0, break_flags: BitField[LineBreakFlag] = 3) const

Breaks text to the lines and returns character ranges for each line.

PackedInt32Array **shaped_text_get_line_breaks_adv**(shaped: RID, width: PackedFloat32Array, start: int = 0, once: bool = true, break_flags: BitField[LineBreakFlag] = 3) const

Breaks text to the lines and columns. Returns character ranges for each segment.

---

int **shaped_text_get_object_glyph**(shaped: RID, key: Variant) const

Returns the glyph index of the inline object.

Vector2i **shaped_text_get_object_range**(shaped: RID, key: Variant) const

Returns the character range of the inline object.

Rect2 **shaped_text_get_object_rect**(shaped: RID, key: Variant) const

Returns bounding rectangle of the inline object.

Array **shaped_text_get_objects**(shaped: RID) const

Returns array of inline objects.

Orientation **shaped_text_get_orientation**(shaped: RID) const

Returns text orientation.

RID **shaped_text_get_parent**(shaped: RID) const

Returns the parent buffer from which the substring originates.

bool **shaped_text_get_preserve_control**(shaped: RID) const

Returns `true` if text buffer is configured to display control characters.

bool **shaped_text_get_preserve_invalid**(shaped: RID) const

Returns `true` if text buffer is configured to display hexadecimal codes in place of invalid characters.

Note: If set to `false`, nothing is displayed in place of invalid characters.

Vector2i **shaped_text_get_range**(shaped: RID) const

Returns substring buffer character range in the parent buffer.

PackedVector2Array **shaped_text_get_selection**(shaped: RID, start: int, end: int) const

Returns selection rectangles for the specified character range.

Vector2 **shaped_text_get_size**(shaped: RID) const

Returns size of the text.

---

int **shaped_text_get_spacing**(shaped: RID, spacing: SpacingType) const

Returns extra spacing added between glyphs or lines in pixels.

---

int **shaped_text_get_trim_pos**(shaped: RID) const

Returns the position of the overrun trim.

---

float **shaped_text_get_underline_position**(shaped: RID) const

Returns pixel offset of the underline below the baseline.

---

float **shaped_text_get_underline_thickness**(shaped: RID) const

Returns thickness of the underline.

---

float **shaped_text_get_width**(shaped: RID) const

Returns width (for horizontal layout) or height (for vertical) of the text.

PackedInt32Array **shaped_text_get_word_breaks**(shaped: RID, grapheme_flags: BitField[GraphemeFlag] = 264, skip_grapheme_flags: BitField[GraphemeFlag] = 4) const

Breaks text into words and returns array of character ranges. Use `grapheme_flags` to set what characters are used for breaking (see GraphemeFlag).

---

bool **shaped_text_has_visible_chars**(shaped: RID) const

Returns `true` if text buffer contains any visible characters.

---

int **shaped_text_hit_test_grapheme**(shaped: RID, coords: float) const 🔗

Returns grapheme index at the specified pixel offset at the baseline, or `-1` if none is found.

---

int **shaped_text_hit_test_position**(shaped: RID, coords: float) const 🔗

Returns caret character offset at the specified pixel offset at the baseline. This function always returns a valid position.

---

bool **shaped_text_is_ready**(shaped: RID) const 🔗

Returns `true` if buffer is successfully shaped.

---

int **shaped_text_next_character_pos**(shaped: RID, pos: int) const 🔗

Returns composite character end position closest to the `pos`.

---

int **shaped_text_next_grapheme_pos**(shaped: RID, pos: int) const 🔗

Returns grapheme end position closest to the `pos`.

---

void **shaped_text_overrun_trim_to_width**(shaped: RID, width: float = 0, overrun_trim_flags: BitField[TextOverrunFlag] = 0) 🔗

Trims text if it exceeds the given width.

---

int **shaped_text_prev_character_pos**(shaped: RID, pos: int) const 🔗

Returns composite character start position closest to the `pos`.

---

int **shaped_text_prev_grapheme_pos**(shaped: RID, pos: int) const 🔗

Returns grapheme start position closest to the `pos`.

---

bool **shaped_text_resize_object**(shaped: RID, key: Variant, size: Vector2, inline_align: InlineAlignment = 5, baseline: float = 0.0)

Sets new size and alignment of embedded object.

---

void **shaped_text_set_bidi_override**(shaped: RID, override: Array)

Overrides BiDi for the structured text.

Override ranges should cover full source text without overlaps. BiDi algorithm will be used on each range separately.

---

void **shaped_text_set_custom_ellipsis**(shaped: RID, char: int)

Sets ellipsis character used for text clipping.

---

void **shaped_text_set_custom_punctuation**(shaped: RID, punct: String)

Sets custom punctuation character list, used for word breaking. If set to empty string, server defaults are used.

---

void **shaped_text_set_direction**(shaped: RID, direction: Direction = 0)

Sets desired text direction. If set to DIRECTION_AUTO, direction will be detected based on the buffer contents and current locale.

Note: Direction is ignored if server does not support FEATURE_BIDI_LAYOUT feature (supported by TextServerAdvanced).

---

void **shaped_text_set_orientation**(shaped: RID, orientation: Orientation = 0)

Sets desired text orientation.

Note: Orientation is ignored if server does not support FEATURE_VERTICAL_LAYOUT feature (supported by TextServerAdvanced).

void **shaped_text_set_preserve_control**(shaped: RID, enabled: bool)

If set to `true` text buffer will display control characters.

void **shaped_text_set_preserve_invalid**(shaped: RID, enabled: bool)

If set to `true` text buffer will display invalid characters as hexadecimal codes, otherwise nothing is displayed.

void **shaped_text_set_spacing**(shaped: RID, spacing: SpacingType, value: int)

Sets extra spacing added between glyphs or lines in pixels.

bool **shaped_text_shape**(shaped: RID)

Shapes buffer if it's not shaped. Returns `true` if the string is shaped successfully.

Note: It is not necessary to call this function manually, buffer will be shaped automatically as soon as any of its output data is requested.

Array[Dictionary] **shaped_text_sort_logical**(shaped: RID)

Returns text glyphs in the logical order.

RID **shaped_text_substr**(shaped: RID, start: int, length: int) const

Returns text buffer for the substring of the text in the `shaped` text buffer (including inline objects).

---

float **shaped_text_tab_align**(shaped: RID, tab_stops: PackedFloat32Array) 🔗

Aligns shaped text to the given tab-stops.

---

bool **spoof_check**(string: String) const 🔗

Returns `true` if `string` is likely to be an attempt at confusing the reader.

Note: Always returns `false` if the server does not support the FEATURE_UNICODE_SECURITY feature.

PackedInt32Array **string_get_character_breaks**(string: String, language: String = "") const 🔗

Returns array of the composite character boundaries.

```
var ts = TextServerManager.get_primary_interface()
print(ts.string_get_word_breaks("Test ❤️ Test")) # Prints [1, 2, 3, 4, 5, 9, 10, 11, 12, 13, 14]
```

PackedInt32Array **string_get_word_breaks**(string: String, language: String = "", chars_per_line: int = 0) const 🔗

Returns an array of the word break boundaries. Elements in the returned array are the offsets of the start and end of words. Therefore the length of the array is always even.

When `chars_per_line` is greater than zero, line break boundaries are returned instead.

```
var ts = TextServerManager.get_primary_interface()
print(ts.string_get_word_breaks("The Godot Engine, 4")) # Prints [0, 3, 4, 9, 10, 16, 18, 19], which corresponds to the following substrings: "The", "Godot", "Engine", "4"
print(ts.string_get_word_breaks("The Godot Engine, 4", "en", 5)) # Prints [0, 3, 4, 9, 10, 15, 15, 19], which corresponds to the following substrings: "The", "Godot", "Engin", "e, 4"
print(ts.string_get_word_breaks("The Godot Engine, 4", "en", 10)) #
```

```
Prints [0, 9, 10, 19], which corresponds to the following substrings:
"The Godot", "Engine, 4"
```

String **string_to_lower**(string: String, language: String = "") const

Returns the string converted to lowercase.

Note: Casing is locale dependent and context sensitive if server support FEATURE_CONTEXT_SENSITIVE_CASE_CONVERSION feature (supported by TextServerAdvanced).

Note: The result may be longer or shorter than the original.

String **string_to_title**(string: String, language: String = "") const

Returns the string converted to title case.

Note: Casing is locale dependent and context sensitive if server support FEATURE_CONTEXT_SENSITIVE_CASE_CONVERSION feature (supported by TextServerAdvanced).

Note: The result may be longer or shorter than the original.

String **string_to_upper**(string: String, language: String = "") const

Returns the string converted to uppercase.

Note: Casing is locale dependent and context sensitive if server support FEATURE_CONTEXT_SENSITIVE_CASE_CONVERSION feature (supported by TextServerAdvanced).

Note: The result may be longer or shorter than the original.

String **strip_diacritics**(string: String) const

Strips diacritics from the string.

Note: The result may be longer or shorter than the original.

String **tag_to_name**(tag: int) const

Converts OpenType tag to readable feature, variation, script, or language name.

TextServerAdvanced

Inherits: TextServerExtension **<** TextServer **<** RefCounted **<** Object

An advanced text server with support for BiDi, complex text layout, and contextual OpenType features. Used in Godot by default.

Description

An implementation of TextServer that uses HarfBuzz, ICU and SIL Graphite to support BiDi, complex text layouts and contextual OpenType features. This is Godot's default primary TextServer interface.

TextServerDummy

Inherits: TextServerExtension **<** TextServer **<** RefCounted **<** Object

A dummy text server that can't render text or manage fonts.

Description

A dummy TextServer interface that doesn't do anything. Useful for freeing up memory when rendering text is not needed, as text servers are resource-intensive. It can also be used for performance comparisons in complex GUIs to check the impact of text rendering.

A dummy text server is always available at the start of a project. Here's how to access it:

```
var dummy_text_server = TextServerManager.find_interface("Dummy")
if dummy_text_server != null:
    TextServerManager.set_primary_interface(dummy_text_server)
    # If the other text servers are unneeded, they can be removed:
    for i in TextServerManager.get_interface_count():
        var text_server = TextServerManager.get_interface(i)
        if text_server != dummy_text_server:
            TextServerManager.remove_interface(text_server)
```

The command line argument `--text-driver Dummy` (case-sensitive) can be used to force the "Dummy" TextServer on any project.

TextServerExtension

Inherits: TextServer < RefCounted < Object

Inherited By: TextServerAdvanced, TextServerDummy, TextServerFallback

Base class for custom TextServer implementations (plugins).

Description

External TextServer implementations should inherit from this class.

Methods

void	_cleanup() virtual
RID	_create_font() virtual
RID	_create_font_linked_variation(font_rid: RID) virtual
RID	_create_shaped_text(direction: Direction, orientation: Orientation) virtual
void	_draw_hex_code_box(canvas: RID, size: int, pos: Vector2, index: int, color: Color) virtual const
void	_font_clear_glyphs(font_rid: RID, size: Vector2i) virtual
void	_font_clear_kerning_map(font_rid: RID, size: int) virtual

void	_font_clear_size_cache(font_rid: RID) virtual
void	_font_clear_textures(font_rid: RID, size: Vector2i) virtual
void	_font_draw_glyph(font_rid: RID, canvas: RID, size: int, pos: Vector2, index: int, color: Color) virtual const
void	_font_draw_glyph_outline(font_rid: RID, canvas: RID, size: int, outline_size: int, pos: Vector2, index: int, color: Color) virtual const
FontAntialiasing	_font_get_antialiasing(font_rid: RID) virtual const
float	_font_get_ascent(font_rid: RID, size: int) virtual const
float	_font_get_baseline_offset(font_rid: RID) virtual const
int	_font_get_char_from_glyph_index(font_rid: RID, size: int, glyph_index: int) virtual const
float	_font_get_descent(font_rid: RID, size: int) virtual const
bool	_font_get_disable_embedded_bitmaps(font_rid: RID) virtual const
float	_font_get_embolden(font_rid: RID) virtual const
int	_font_get_face_count(font_rid: RID) virtual const
int	_font_get_face_index(font_rid: RID) virtual const

int	_font_get_fixed_size(font_rid: RID) virtual const
FixedSizeScaleMode	_font_get_fixed_size_scale_mode(font_rid: RID) virtual const
bool	_font_get_generate_mipmaps(font_rid: RID) virtual const
float	_font_get_global_oversampling() virtual const
Vector2	_font_get_glyph_advance(font_rid: RID, size: int, glyph: int) virtual const
Dictionary	_font_get_glyph_contours(font_rid: RID, size: int, index: int) virtual const
int	_font_get_glyph_index(font_rid: RID, size: int, char: int, variation_selector: int) virtual const
PackedInt32Array	_font_get_glyph_list(font_rid: RID, size: Vector2i) virtual const
Vector2	_font_get_glyph_offset(font_rid: RID, size: Vector2i, glyph: int) virtual const
Vector2	_font_get_glyph_size(font_rid: RID, size: Vector2i, glyph: int) virtual const
int	_font_get_glyph_texture_idx(font_rid: RID, size: Vector2i, glyph: int) virtual const

RID	_font_get_glyph_texture_rid(font_rid: RID, size: Vector2i, glyph: int) virtual const
Vector2	_font_get_glyph_texture_size(font_rid: RID, size: Vector2i, glyph: int) virtual const
Rect2	_font_get_glyph_uv_rect(font_rid: RID, size: Vector2i, glyph: int) virtual const
Hinting	_font_get_hinting(font_rid: RID) virtual const
Vector2	_font_get_kerning(font_rid: RID, size: int, glyph_pair: Vector2i) virtual const
Array[Vector2i]	_font_get_kerning_list(font_rid: RID, size: int) virtual const
bool	_font_get_language_support_override(font_rid: RID, language: String) virtual
PackedStringArray	_font_get_language_support_overrides(font_rid: RID) virtual
int	_font_get_msdf_pixel_range(font_rid: RID) virtual const
int	_font_get_msdf_size(font_rid: RID) virtual const
String	_font_get_name(font_rid: RID) virtual const
Dictionary	_font_get_opentype_feature_overrides(font_rid: RID) virtual const

Dictionary	_font_get_ot_name_strings(font_rid: RID) virtual const
float	_font_get_oversampling(font_rid: RID) virtual const
float	_font_get_scale(font_rid: RID, size: int) virtual const
bool	_font_get_script_support_override(font_rid: RID, script: String) virtual
PackedStringArray	_font_get_script_support_overrides(font_rid: RID) virtual
Array[Vector2i]	_font_get_size_cache_list(font_rid: RID) virtual const
int	_font_get_spacing(font_rid: RID, spacing: SpacingType) virtual const
int	_font_get_stretch(font_rid: RID) virtual const
BitField[FontStyle]	_font_get_style(font_rid: RID) virtual const
String	_font_get_style_name(font_rid: RID) virtual const
SubpixelPositioning	_font_get_subpixel_positioning(font_rid: RID) virtual const
String	_font_get_supported_chars(font_rid: RID) virtual const
int	_font_get_texture_count(font_rid: RID, size: Vector2i) virtual const

Image	_font_get_texture_image(font_rid: RID, size: Vector2i, texture_index: int) virtual const
PackedInt32Array	_font_get_texture_offsets(font_rid: RID, size: Vector2i, texture_index: int) virtual const
Transform2D	_font_get_transform(font_rid: RID) virtual const
float	_font_get_underline_position(font_rid: RID, size: int) virtual const
float	_font_get_underline_thickness(font_rid: RID, size: int) virtual const
Dictionary	_font_get_variation_coordinates(font_rid: RID) virtual const
int	_font_get_weight(font_rid: RID) virtual const
bool	_font_has_char(font_rid: RID, char: int) virtual const
bool	_font_is_allow_system_fallback(font_rid: RID) virtual const
bool	_font_is_force_autohinter(font_rid: RID) virtual const
bool	_font_is_language_supported(font_rid: RID, language: String) virtual const

bool	_font_is_multichannel_signed_distance_field(font_rid: RID) virtual const
bool	_font_is_script_supported(font_rid: RID, script: String) virtual const
void	_font_remove_glyph(font_rid: RID, size: Vector2i, glyph: int) virtual
void	_font_remove_kerning(font_rid: RID, size: int, glyph_pair: Vector2i) virtual
void	_font_remove_language_support_override(font_rid: RID, language: String) virtual
void	_font_remove_script_support_override(font_rid: RID, script: String) virtual
void	_font_remove_size_cache(font_rid: RID, size: Vector2i) virtual
void	_font_remove_texture(font_rid: RID, size: Vector2i, texture_index: int) virtual
void	_font_render_glyph(font_rid: RID, size: Vector2i, index: int) virtual
void	_font_render_range(font_rid: RID, size: Vector2i, start: int, end: int) virtual
void	_font_set_allow_system_fallback(font_rid: RID, allow_system_fallback: bool) virtual

void	_font_set_antialiasing(font_rid: RID, antialiasing: FontAntialiasing) virtual
void	_font_set_ascent(font_rid: RID, size: int, ascent: float) virtual
void	_font_set_baseline_offset(font_rid: RID, baseline_offset: float) virtual
void	_font_set_data(font_rid: RID, data: PackedByteArray) virtual
void	_font_set_data_ptr(font_rid: RID, data_ptr: `const uint8_t*`, data_size: int) virtual
void	_font_set_descent(font_rid: RID, size: int, descent: float) virtual
void	_font_set_disable_embedded_bitmaps(font_rid: RID, disable_embedded_bitmaps: bool) virtual
void	_font_set_embolden(font_rid: RID, strength: float) virtual
void	_font_set_face_index(font_rid: RID, face_index: int) virtual
void	_font_set_fixed_size(font_rid: RID, fixed_size: int) virtual

void	_font_set_fixed_size_scale_mode(font_rid: RID, fixed_size_scale_mode: FixedSizeScaleMode) virtual
void	_font_set_force_autohinter(font_rid: RID, force_autohinter: bool) virtual
void	_font_set_generate_mipmaps(font_rid: RID, generate_mipmaps: bool) virtual
void	_font_set_global_oversampling(oversampling: float) virtual
void	_font_set_glyph_advance(font_rid: RID, size: int, glyph: int, advance: Vector2) virtual
void	_font_set_glyph_offset(font_rid: RID, size: Vector2i, glyph: int, offset: Vector2) virtual
void	_font_set_glyph_size(font_rid: RID, size: Vector2i, glyph: int, gl_size: Vector2) virtual
void	_font_set_glyph_texture_idx(font_rid: RID, size: Vector2i, glyph: int, texture_idx: int) virtual
void	_font_set_glyph_uv_rect(font_rid: RID, size: Vector2i, glyph: int, uv_rect: Rect2) virtual
void	_font_set_hinting(font_rid: RID, hinting: Hinting) virtual
void	_font_set_kerning(font_rid: RID, size: int, glyph_pair: Vector2i, kerning: Vector2) virtual

void	_font_set_language_support_override(font_rid: RID, language: String, supported: bool) virtual
void	_font_set_msdf_pixel_range(font_rid: RID, msdf_pixel_range: int) virtual
void	_font_set_msdf_size(font_rid: RID, msdf_size: int) virtual
void	_font_set_multichannel_signed_distance_field(font_rid: RID, msdf: bool) virtual
void	_font_set_name(font_rid: RID, name: String) virtual
void	_font_set_opentype_feature_overrides(font_rid: RID, overrides: Dictionary) virtual
void	_font_set_oversampling(font_rid: RID, oversampling: float) virtual
void	_font_set_scale(font_rid: RID, size: int, scale: float) virtual
void	_font_set_script_support_override(font_rid: RID, script: String, supported: bool) virtual
void	_font_set_spacing(font_rid: RID, spacing: SpacingType, value: int) virtual
void	_font_set_stretch(font_rid: RID, stretch: int) virtual

void	_font_set_style(font_rid: RID, style: BitField[FontStyle]) virtual
void	_font_set_style_name(font_rid: RID, name_style: String) virtual
void	_font_set_subpixel_positioning(font_rid: RID, subpixel_positioning: SubpixelPositioning) virtual
void	_font_set_texture_image(font_rid: RID, size: Vector2i, texture_index: int, image: Image) virtual
void	_font_set_texture_offsets(font_rid: RID, size: Vector2i, texture_index: int, offset: PackedInt32Array) virtual
void	_font_set_transform(font_rid: RID, transform: Transform2D) virtual
void	_font_set_underline_position(font_rid: RID, size: int, underline_position: float) virtual
void	_font_set_underline_thickness(font_rid: RID, size: int, underline_thickness: float) virtual
void	_font_set_variation_coordinates(font_rid: RID, variation_coordinates: Dictionary) virtual
void	_font_set_weight(font_rid: RID, weight: int) virtual
Dictionary	_font_supported_feature_list(font_rid: RID) virtual const

Dictionary	_font_supported_variation_list(font_rid: RID) virtual const
String	_format_number(number: String, language: String) virtual const
void	_free_rid(rid: RID) virtual
int	_get_features() virtual const
Vector2	_get_hex_code_box_size(size: int, index: int) virtual const
String	_get_name() virtual const
String	_get_support_data_filename() virtual const
String	_get_support_data_info() virtual const
bool	_has(rid: RID) virtual
bool	_has_feature(feature: Feature) virtual const
int	_is_confusable(string: String, dict: PackedStringArray) virtual const
bool	_is_locale_right_to_left(locale: String) virtual const
bool	_is_valid_identifier(string: String) virtual const

bool	_is_valid_letter(unicode: int) virtual const
bool	_load_support_data(filename: String) virtual
int	_name_to_tag(name: String) virtual const
String	_parse_number(number: String, language: String) virtual const
Array[Vector3i]	_parse_structured_text(parser_type: StructuredTextParser, args: Array, text: String) virtual const
String	_percent_sign(language: String) virtual const
bool	_save_support_data(filename: String) virtual const
int	_shaped_get_span_count(shaped: RID) virtual const
Variant	_shaped_get_span_meta(shaped: RID, index: int) virtual const
void	_shaped_set_span_update_font(shaped: RID, index: int, fonts: Array[RID], size: int, opentype_features: Dictionary) virtual
bool	_shaped_text_add_object(shaped: RID, key: Variant, size: Vector2, inline_align: InlineAlignment, length: int, baseline: float) virtual

bool	_shaped_text_add_string(shaped: RID, text: String, fonts: Array[RID], size: int, opentype_features: Dictionary, language: String, meta: Variant) virtual
void	_shaped_text_clear(shaped: RID) virtual
int	_shaped_text_closest_character_pos(shaped: RID, pos: int) virtual const
void	_shaped_text_draw(shaped: RID, canvas: RID, pos: Vector2, clip_l: float, clip_r: float, color: Color) virtual const
void	_shaped_text_draw_outline(shaped: RID, canvas: RID, pos: Vector2, clip_l: float, clip_r: float, outline_size: int, color: Color) virtual const
float	_shaped_text_fit_to_width(shaped: RID, width: float, justification_flags: BitField[JustificationFlag]) virtual
float	_shaped_text_get_ascent(shaped: RID) virtual const
void	_shaped_text_get_carets(shaped: RID, position: int, caret: `CaretInfo*`) virtual const
PackedInt32Array	_shaped_text_get_character_breaks(shaped: RID) virtual const
int	_shaped_text_get_custom_ellipsis(shaped: RID) virtual const

String	_shaped_text_get_custom_punctuation(shaped: RID) virtual const
float	_shaped_text_get_descent(shaped: RID) virtual const
Direction	_shaped_text_get_direction(shaped: RID) virtual const
int	_shaped_text_get_dominant_direction_in_range(shaped: RID, start: int, end: int) virtual const
int	_shaped_text_get_ellipsis_glyph_count(shaped: RID) virtual const
`const Glyph*`	_shaped_text_get_ellipsis_glyphs(shaped: RID) virtual const
int	_shaped_text_get_ellipsis_pos(shaped: RID) virtual const
int	_shaped_text_get_glyph_count(shaped: RID) virtual const
`const Glyph*`	_shaped_text_get_glyphs(shaped: RID) virtual const
Vector2	_shaped_text_get_grapheme_bounds(shaped: RID, pos: int) virtual const
Direction	_shaped_text_get_inferred_direction(shaped: RID) virtual const

PackedInt32Array	_shaped_text_get_line_breaks(shaped: RID, width: float, start: int, break_flags: BitField[LineBreakFlag]) virtual const
PackedInt32Array	_shaped_text_get_line_breaks_adv(shaped: RID, width: PackedFloat32Array, start: int, once: bool, break_flags: BitField[LineBreakFlag]) virtual const
int	_shaped_text_get_object_glyph(shaped: RID, key: Variant) virtual const
Vector2i	_shaped_text_get_object_range(shaped: RID, key: Variant) virtual const
Rect2	_shaped_text_get_object_rect(shaped: RID, key: Variant) virtual const
Array	_shaped_text_get_objects(shaped: RID) virtual const
Orientation	_shaped_text_get_orientation(shaped: RID) virtual const
RID	_shaped_text_get_parent(shaped: RID) virtual const
bool	_shaped_text_get_preserve_control(shaped: RID) virtual const
bool	_shaped_text_get_preserve_invalid(shaped: RID) virtual const
Vector2i	_shaped_text_get_range(shaped: RID) virtual const

PackedVector2Array	_shaped_text_get_selection(shaped: RID, start: int, end: int) virtual const
Vector2	_shaped_text_get_size(shaped: RID) virtual const
int	_shaped_text_get_spacing(shaped: RID, spacing: SpacingType) virtual const
int	_shaped_text_get_trim_pos(shaped: RID) virtual const
float	_shaped_text_get_underline_position(shaped: RID) virtual const
float	_shaped_text_get_underline_thickness(shaped: RID) virtual const
float	_shaped_text_get_width(shaped: RID) virtual const
PackedInt32Array	_shaped_text_get_word_breaks(shaped: RID, grapheme_flags: BitField[GraphemeFlag], skip_grapheme_flags: BitField[GraphemeFlag]) virtual const
int	_shaped_text_hit_test_grapheme(shaped: RID, coord: float) virtual const
int	_shaped_text_hit_test_position(shaped: RID, coord: float) virtual const
bool	_shaped_text_is_ready(shaped: RID) virtual const

int	_shaped_text_next_character_pos(shaped: RID, pos: int) virtual const
int	_shaped_text_next_grapheme_pos(shaped: RID, pos: int) virtual const
void	_shaped_text_overrun_trim_to_width(shaped: RID, width: float, trim_flags: BitField[TextOverrunFlag]) virtual
int	_shaped_text_prev_character_pos(shaped: RID, pos: int) virtual const
int	_shaped_text_prev_grapheme_pos(shaped: RID, pos: int) virtual const
bool	_shaped_text_resize_object(shaped: RID, key: Variant, size: Vector2, inline_align: InlineAlignment, baseline: float) virtual
void	_shaped_text_set_bidi_override(shaped: RID, override: Array) virtual
void	_shaped_text_set_custom_ellipsis(shaped: RID, char: int) virtual
void	_shaped_text_set_custom_punctuation(shaped: RID, punct: String) virtual
void	_shaped_text_set_direction(shaped: RID, direction: Direction) virtual

void	_shaped_text_set_orientation(shaped: RID, orientation: Orientation) virtual
void	_shaped_text_set_preserve_control(shaped: RID, enabled: bool) virtual
void	_shaped_text_set_preserve_invalid(shaped: RID, enabled: bool) virtual
void	_shaped_text_set_spacing(shaped: RID, spacing: SpacingType, value: int) virtual
bool	_shaped_text_shape(shaped: RID) virtual
`const Glyph*`	_shaped_text_sort_logical(shaped: RID) virtual
RID	_shaped_text_substr(shaped: RID, start: int, length: int) virtual const
float	_shaped_text_tab_align(shaped: RID, tab_stops: PackedFloat32Array) virtual
bool	_shaped_text_update_breaks(shaped: RID) virtual
bool	_shaped_text_update_justification_ops(shaped: RID) virtual
bool	_spoof_check(string: String) virtual const
PackedInt32Array	_string_get_character_breaks(string: String, language: String) virtual const

PackedInt32Array	_string_get_word_breaks(string: String, language: String, chars_per_line: int) virtual const
String	_string_to_lower(string: String, language: String) virtual const
String	_string_to_title(string: String, language: String) virtual const
String	_string_to_upper(string: String, language: String) virtual const
String	_strip_diacritics(string: String) virtual const
String	_tag_to_name(tag: int) virtual const

Method Descriptions

void **_cleanup**() virtual 🔗

Optional.

This method is called before text server is unregistered.

RID **_create_font**() virtual 🔗

Required.

Creates a new, empty font cache entry resource.

RID **_create_font_linked_variation**(font_rid: RID) virtual 🔗

Optional, implement if font supports extra spacing or baseline offset.

Creates a new variation existing font which is reusing the same glyph cache and font data.

RID **_create_shaped_text**(direction: Direction, orientation: Orientation) virtual

Required.

Creates a new buffer for complex text layout, with the given `direction` and `orientation`.

---

void **_draw_hex_code_box**(canvas: RID, size: int, pos: Vector2, index: int, color: Color) virtual const

Optional.

Draws box displaying character hexadecimal code.

---

void **_font_clear_glyphs**(font_rid: RID, size: Vector2i) virtual

Required.

Removes all rendered glyph information from the cache entry.

---

void **_font_clear_kerning_map**(font_rid: RID, size: int) virtual

Optional.

Removes all kerning overrides.

---

void **_font_clear_size_cache**(font_rid: RID) virtual

Required.

Removes all font sizes from the cache entry.

---

void **_font_clear_textures**(font_rid: RID, size: Vector2i) virtual 🔗

Required.

Removes all textures from font cache entry.

---

void **_font_draw_glyph**(font_rid: RID, canvas: RID, size: int, pos: Vector2, index: int, color: Color) virtual const 🔗

Required.

Draws single glyph into a canvas item at the position, using `font_rid` at the size `size`.

---

void **_font_draw_glyph_outline**(font_rid: RID, canvas: RID, size: int, outline_size: int, pos: Vector2, index: int, color: Color) virtual const 🔗

Required.

Draws single glyph outline of size `outline_size` into a canvas item at the position, using `font_rid` at the size `size`.

FontAntialiasing **_font_get_antialiasing**(font_rid: RID) virtual const 🔗

Optional.

Returns font anti-aliasing mode.

---

float **_font_get_ascent**(font_rid: RID, size: int) virtual const 🔗

Required.

Returns the font ascent (number of pixels above the baseline).

---

float **_font_get_baseline_offset**(font_rid: RID) virtual const 🔗

Optional.

Returns extra baseline offset (as a fraction of font height).

---

int **_font_get_char_from_glyph_index**(font_rid: RID, size: int, glyph_index: int) virtual const 🔗

Required.

Returns character code associated with `glyph_index`, or `0` if `glyph_index` is invalid.

---

float **_font_get_descent**(font_rid: RID, size: int) virtual const 🔗

Required.

Returns the font descent (number of pixels below the baseline).

---

bool **_font_get_disable_embedded_bitmaps**(font_rid: RID) virtual const 🔗

Optional.

Returns whether the font's embedded bitmap loading is disabled.

---

float **_font_get_embolden**(font_rid: RID) virtual const 🔗

Optional.

Returns font embolden strength.

---

int **_font_get_face_count**(font_rid: RID) virtual const 🔗

Optional.

Returns number of faces in the TrueType / OpenType collection.

---

int **_font_get_face_index**(font_rid: RID) virtual const 🔗

Optional.

Returns an active face index in the TrueType / OpenType collection.

int **_font_get_fixed_size**(font_rid: RID) virtual const 🔗

Required.

Returns bitmap font fixed size.

FixedSizeScaleMode **_font_get_fixed_size_scale_mode**(font_rid: RID) virtual const 🔗

Required.

Returns bitmap font scaling mode.

bool **_font_get_generate_mipmaps**(font_rid: RID) virtual const 🔗

Optional.

Returns `true` if font texture mipmap generation is enabled.

float **_font_get_global_oversampling**() virtual const 🔗

Optional.

Returns the font oversampling factor, shared by all fonts in the TextServer.

Vector2 **_font_get_glyph_advance**(font_rid: RID, size: int, glyph: int) virtual const 🔗

Required.

Returns glyph advance (offset of the next glyph).

Dictionary **_font_get_glyph_contours**(font_rid: RID, size: int, index: int) virtual const

Optional.

Returns outline contours of the glyph.

int **_font_get_glyph_index**(font_rid: RID, size: int, char: int, variation_selector: int) virtual const

Required.

Returns the glyph index of a `char`, optionally modified by the `variation_selector`.

PackedInt32Array **_font_get_glyph_list**(font_rid: RID, size: Vector2i) virtual const

Required.

Returns list of rendered glyphs in the cache entry.

Vector2 **_font_get_glyph_offset**(font_rid: RID, size: Vector2i, glyph: int) virtual const

Required.

Returns glyph offset from the baseline.

Vector2 **_font_get_glyph_size**(font_rid: RID, size: Vector2i, glyph: int) virtual const

Required.

Returns size of the glyph.

int **_font_get_glyph_texture_idx**(font_rid: RID, size: Vector2i, glyph: int) virtual const

Required.

Returns index of the cache texture containing the glyph.

RID **_font_get_glyph_texture_rid**(font_rid: RID, size: Vector2i, glyph: int) virtual const

Required.

Returns resource ID of the cache texture containing the glyph.

Vector2 **_font_get_glyph_texture_size**(font_rid: RID, size: Vector2i, glyph: int) virtual const

Required.

Returns size of the cache texture containing the glyph.

Rect2 **_font_get_glyph_uv_rect**(font_rid: RID, size: Vector2i, glyph: int) virtual const

Required.

Returns rectangle in the cache texture containing the glyph.

Hinting **_font_get_hinting**(font_rid: RID) virtual const

Optional.

Returns the font hinting mode. Used by dynamic fonts only.

Vector2 **_font_get_kerning**(font_rid: RID, size: int, glyph_pair: Vector2i) virtual const

Optional.

Returns kerning for the pair of glyphs.

Array[Vector2i] **_font_get_kerning_list**(font_rid: RID, size: int) virtual const

Optional.

Returns list of the kerning overrides.

---

bool **_font_get_language_support_override**(font_rid: RID, language: String) virtual

Optional.

Returns `true` if support override is enabled for the `language`.

PackedStringArray **_font_get_language_support_overrides**(font_rid: RID) virtual

Optional.

Returns list of language support overrides.

---

int **_font_get_msdf_pixel_range**(font_rid: RID) virtual const

Optional.

Returns the width of the range around the shape between the minimum and maximum representable signed distance.

---

int **_font_get_msdf_size**(font_rid: RID) virtual const

Optional.

Returns source font size used to generate MSDF textures.

String **_font_get_name**(font_rid: RID) virtual const

Optional.

Returns font family name.

Dictionary **_font_get_opentype_feature_overrides**(font_rid: RID) virtual const

Optional.

Returns font OpenType feature set override.

Dictionary **_font_get_ot_name_strings**(font_rid: RID) virtual const

Optional.

Returns Dictionary with OpenType font name strings (localized font names, version, description, license information, sample text, etc.).

float **_font_get_oversampling**(font_rid: RID) virtual const

Optional.

Returns font oversampling factor, if set to `0.0` global oversampling factor is used instead. Used by dynamic fonts only.

float **_font_get_scale**(font_rid: RID, size: int) virtual const

Required.

Returns scaling factor of the color bitmap font.

bool **_font_get_script_support_override**(font_rid: RID, script: String) virtual

Optional.

Returns `true` if support override is enabled for the `script`.

PackedStringArray **_font_get_script_support_overrides**(font_rid: RID) virtual

Optional.

Returns list of script support overrides.

Array[Vector2i] **_font_get_size_cache_list**(font_rid: RID) virtual const

Required.

Returns list of the font sizes in the cache. Each size is Vector2i with font size and outline size.

---

int **_font_get_spacing**(font_rid: RID, spacing: SpacingType) virtual const

Optional.

Returns the spacing for `spacing` (see SpacingType) in pixels (not relative to the font size).

---

int **_font_get_stretch**(font_rid: RID) virtual const

Optional.

Returns font stretch amount, compared to a normal width. A percentage value between `50%` and `200%`.

BitField[FontStyle] **_font_get_style**(font_rid: RID) virtual const

Optional.

Returns font style flags, see FontStyle.

String **_font_get_style_name**(font_rid: RID) virtual const 🔗

Optional.

Returns font style name.

SubpixelPositioning **_font_get_subpixel_positioning**(font_rid: RID) virtual const 🔗

Optional.

Returns font subpixel glyph positioning mode.

String **_font_get_supported_chars**(font_rid: RID) virtual const 🔗

Required.

Returns a string containing all the characters available in the font.

---

int **_font_get_texture_count**(font_rid: RID, size: Vector2i) virtual const 🔗

Required.

Returns number of textures used by font cache entry.

Image **_font_get_texture_image**(font_rid: RID, size: Vector2i, texture_index: int) virtual const 🔗

Required.

Returns font cache texture image data.

PackedInt32Array **_font_get_texture_offsets**(font_rid: RID, size: Vector2i, texture_index: int) virtual const 🔗

Optional.

Returns array containing glyph packing data.

Transform2D **_font_get_transform**(font_rid: RID) virtual const

Optional.

Returns 2D transform applied to the font outlines.

---

float **_font_get_underline_position**(font_rid: RID, size: int) virtual const

Required.

Returns pixel offset of the underline below the baseline.

---

float **_font_get_underline_thickness**(font_rid: RID, size: int) virtual const

Required.

Returns thickness of the underline in pixels.

Dictionary **_font_get_variation_coordinates**(font_rid: RID) virtual const

Optional.

Returns variation coordinates for the specified font cache entry.

---

int **_font_get_weight**(font_rid: RID) virtual const

Optional.

Returns weight (boldness) of the font. A value in the `100...999` range, normal font weight is `400`, bold font weight is `700`.

---

bool **_font_has_char**(font_rid: RID, char: int) virtual const

Required.

Returns `true` if a Unicode `char` is available in the font.

bool **_font_is_allow_system_fallback**(font_rid: RID) virtual const

Optional.

Returns `true` if system fonts can be automatically used as fallbacks.

bool **_font_is_force_autohinter**(font_rid: RID) virtual const

Optional.

Returns `true` if auto-hinting is supported and preferred over font built-in hinting.

bool **_font_is_language_supported**(font_rid: RID, language: String) virtual const

Optional.

Returns `true`, if font supports given language (ISO 639 [https://en.wikipedia.org/wiki/ISO_639-1] code).

bool **_font_is_multichannel_signed_distance_field**(font_rid: RID) virtual const

Optional.

Returns `true` if glyphs of all sizes are rendered using single multichannel signed distance field generated from the dynamic font vector data.

bool **_font_is_script_supported**(font_rid: RID, script: String) virtual const

Optional.

Returns `true`, if font supports given script (ISO 15924 code).

---

void **_font_remove_glyph**(font_rid: RID, size: Vector2i, glyph: int) virtual

Required.

Removes specified rendered glyph information from the cache entry.

---

void **_font_remove_kerning**(font_rid: RID, size: int, glyph_pair: Vector2i) virtual

Optional.

Removes kerning override for the pair of glyphs.

---

void **_font_remove_language_support_override**(font_rid: RID, language: String) virtual

Optional.

Remove language support override.

---

void **_font_remove_script_support_override**(font_rid: RID, script: String) virtual

Optional.

Removes script support override.

---

void **_font_remove_size_cache**(font_rid: RID, size: Vector2i) virtual

Required.

Removes specified font size from the cache entry.

---

void **_font_remove_texture**(font_rid: RID, size: Vector2i, texture_index: int) virtual

Required.

Removes specified texture from the cache entry.

---

void **_font_render_glyph**(font_rid: RID, size: Vector2i, index: int) virtual

Optional.

Renders specified glyph to the font cache texture.

---

void **_font_render_range**(font_rid: RID, size: Vector2i, start: int, end: int) virtual

Optional.

Renders the range of characters to the font cache texture.

---

void **_font_set_allow_system_fallback**(font_rid: RID, allow_system_fallback: bool) virtual

Optional.

If set to `true`, system fonts can be automatically used as fallbacks.

---

void **_font_set_antialiasing**(font_rid: RID, antialiasing: FontAntialiasing) virtual

Optional.

Sets font anti-aliasing mode.

---

void **_font_set_ascent**(font_rid: RID, size: int, ascent: float) virtual

Required.

Sets the font ascent (number of pixels above the baseline).

void **_font_set_baseline_offset**(font_rid: RID, baseline_offset: float) virtual 🔗

Optional.

Sets extra baseline offset (as a fraction of font height).

void **_font_set_data**(font_rid: RID, data: PackedByteArray) virtual 🔗

Optional.

Sets font source data, e.g contents of the dynamic font source file.

void **_font_set_data_ptr**(font_rid: RID, data_ptr: `const uint8_t*`, data_size: int) virtual 🔗

Optional.

Sets pointer to the font source data, e.g contents of the dynamic font source file.

void **_font_set_descent**(font_rid: RID, size: int, descent: float) virtual 🔗

Required.

Sets the font descent (number of pixels below the baseline).

void **_font_set_disable_embedded_bitmaps**(font_rid: RID, disable_embedded_bitmaps: bool) virtual 🔗

Optional.

If set to `true`, embedded font bitmap loading is disabled.

void **_font_set_embolden**(font_rid: RID, strength: float) virtual

Sets font embolden strength. If `strength` is not equal to zero, emboldens the font outlines. Negative values reduce the outline thickness.

void **_font_set_face_index**(font_rid: RID, face_index: int) virtual

Optional.

Sets an active face index in the TrueType / OpenType collection.

void **_font_set_fixed_size**(font_rid: RID, fixed_size: int) virtual

Required.

Sets bitmap font fixed size. If set to value greater than zero, same cache entry will be used for all font sizes.

void **_font_set_fixed_size_scale_mode**(font_rid: RID, fixed_size_scale_mode: FixedSizeScaleMode) virtual

Required.

Sets bitmap font scaling mode. This property is used only if `fixed_size` is greater than zero.

void **_font_set_force_autohinter**(font_rid: RID, force_autohinter: bool) virtual

Optional.

If set to `true` auto-hinting is preferred over font built-in hinting.

void **_font_set_generate_mipmaps**(font_rid: RID, generate_mipmaps: bool) virtual

Optional.

If set to `true` font texture mipmap generation is enabled.

void **_font_set_global_oversampling**(oversampling: float) virtual

Optional.

Sets oversampling factor, shared by all font in the TextServer.

void **_font_set_glyph_advance**(font_rid: RID, size: int, glyph: int, advance: Vector2) virtual

Required.

Sets glyph advance (offset of the next glyph).

void **_font_set_glyph_offset**(font_rid: RID, size: Vector2i, glyph: int, offset: Vector2) virtual

Required.

Sets glyph offset from the baseline.

void **_font_set_glyph_size**(font_rid: RID, size: Vector2i, glyph: int, gl_size: Vector2) virtual

Required.

Sets size of the glyph.

void **_font_set_glyph_texture_idx**(font_rid: RID, size: Vector2i, glyph: int, texture_idx: int) virtual

Required.

Sets index of the cache texture containing the glyph.

void **_font_set_glyph_uv_rect**(font_rid: RID, size: Vector2i, glyph: int, uv_rect: Rect2) virtual 🔗

Required.

Sets rectangle in the cache texture containing the glyph.

void **_font_set_hinting**(font_rid: RID, hinting: Hinting) virtual 🔗

Optional.

Sets font hinting mode. Used by dynamic fonts only.

void **_font_set_kerning**(font_rid: RID, size: int, glyph_pair: Vector2i, kerning: Vector2) virtual 🔗

Optional.

Sets kerning for the pair of glyphs.

void **_font_set_language_support_override**(font_rid: RID, language: String, supported: bool) virtual 🔗

Optional.

Adds override for _font_is_language_supported.

void **_font_set_msdf_pixel_range**(font_rid: RID, msdf_pixel_range: int) virtual 🔗

Optional.

Sets the width of the range around the shape between the minimum and maximum representable signed distance.

void **_font_set_msdf_size**(font_rid: RID, msdf_size: int) virtual 🔗

Optional.

Sets source font size used to generate MSDF textures.

---

void **_font_set_multichannel_signed_distance_field**(font_rid: RID, msdf: bool) virtual 🔗

Optional.

If set to `true`, glyphs of all sizes are rendered using single multichannel signed distance field generated from the dynamic font vector data. MSDF rendering allows displaying the font at any scaling factor without blurriness, and without incurring a CPU cost when the font size changes (since the font no longer needs to be rasterized on the CPU). As a downside, font hinting is not available with MSDF. The lack of font hinting may result in less crisp and less readable fonts at small sizes.

---

void **_font_set_name**(font_rid: RID, name: String) virtual 🔗

Optional.

Sets the font family name.

---

void **_font_set_opentype_feature_overrides**(font_rid: RID, overrides: Dictionary) virtual 🔗

Optional.

Sets font OpenType feature set override.

---

void **_font_set_oversampling**(font_rid: RID, oversampling: float) virtual 🔗

Optional.

Sets font oversampling factor, if set to `0.0` global oversampling factor is used instead. Used by dynamic fonts only.

---

void **_font_set_scale**(font_rid: RID, size: int, scale: float) virtual 🔗

Required.

Sets scaling factor of the color bitmap font.

---

void **_font_set_script_support_override**(font_rid: RID, script: String, supported: bool) virtual

Optional.

Adds override for _font_is_script_supported.

---

void **_font_set_spacing**(font_rid: RID, spacing: SpacingType, value: int) virtual

Optional.

Sets the spacing for `spacing` (see SpacingType) to `value` in pixels (not relative to the font size).

---

void **_font_set_stretch**(font_rid: RID, stretch: int) virtual

Optional.

Sets font stretch amount, compared to a normal width. A percentage value between `50%` and `200%`.

---

void **_font_set_style**(font_rid: RID, style: BitField[FontStyle]) virtual

Optional.

Sets the font style flags, see FontStyle.

---

void **_font_set_style_name**(font_rid: RID, name_style: String) virtual

Optional.

Sets the font style name.

---

void **_font_set_subpixel_positioning**(font_rid: RID, subpixel_positioning: SubpixelPositioning) virtual

Optional.

Sets font subpixel glyph positioning mode.

void **_font_set_texture_image**(font_rid: RID, size: Vector2i, texture_index: int, image: Image) virtual

Required.

Sets font cache texture image data.

void **_font_set_texture_offsets**(font_rid: RID, size: Vector2i, texture_index: int, offset: PackedInt32Array) virtual

Optional.

Sets array containing glyph packing data.

void **_font_set_transform**(font_rid: RID, transform: Transform2D) virtual

Optional.

Sets 2D transform, applied to the font outlines, can be used for slanting, flipping, and rotating glyphs.

void **_font_set_underline_position**(font_rid: RID, size: int, underline_position: float) virtual

Required.

Sets pixel offset of the underline below the baseline.

void **_font_set_underline_thickness**(font_rid: RID, size: int, underline_thickness: float) virtual

Required.

Sets thickness of the underline in pixels.

void **_font_set_variation_coordinates**(font_rid: RID, variation_coordinates: Dictionary) virtual

Optional.

Sets variation coordinates for the specified font cache entry.

void **_font_set_weight**(font_rid: RID, weight: int) virtual

Optional.

Sets weight (boldness) of the font. A value in the `100...999` range, normal font weight is `400`, bold font weight is `700`.

Dictionary **_font_supported_feature_list**(font_rid: RID) virtual const

Optional.

Returns the dictionary of the supported OpenType features.

Dictionary **_font_supported_variation_list**(font_rid: RID) virtual const

Optional.

Returns the dictionary of the supported OpenType variation coordinates.

String **_format_number**(number: String, language: String) virtual const

Optional.

Converts a number from the Western Arabic (0..9) to the numeral systems used in `language`.

---

void **_free_rid**(rid: RID) virtual

Required.

Frees an object created by this TextServer.

---

int **_get_features**() virtual const

Required.

Returns text server features, see Feature.

Vector2 **_get_hex_code_box_size**(size: int, index: int) virtual const

Optional.

Returns size of the replacement character (box with character hexadecimal code that is drawn in place of invalid characters).

String **_get_name**() virtual const

Required.

Returns the name of the server interface.

String **_get_support_data_filename**() virtual const

Optional.

Returns default TextServer database (e.g. ICU break iterators and dictionaries) filename.

String **_get_support_data_info**() virtual const

Optional.

Returns TextServer database (e.g. ICU break iterators and dictionaries) description.

bool **_has**(rid: RID) virtual

Required.

Returns `true` if `rid` is valid resource owned by this text server.

bool **_has_feature**(feature: Feature) virtual const

Required.

Returns `true` if the server supports a feature.

int **_is_confusable**(string: String, dict: PackedStringArray) virtual const

Optional.

Returns index of the first string in `dict` which is visually confusable with the `string`, or `-1` if none is found.

bool **_is_locale_right_to_left**(locale: String) virtual const

Required.

Returns `true` if locale is right-to-left.

bool **_is_valid_identifier**(string: String) virtual const

Optional.

Returns `true` if `string` is a valid identifier.

bool **_is_valid_letter**(unicode: int) virtual const

There is currently no description for this method. Please help us by contributing one!

---

bool **_load_support_data**(filename: String) virtual

Optional.

Loads optional TextServer database (e.g. ICU break iterators and dictionaries).

---

int **_name_to_tag**(name: String) virtual const

Optional.

Converts readable feature, variation, script, or language name to OpenType tag.

String **_parse_number**(number: String, language: String) virtual const

Optional.

Converts `number` from the numeral systems used in `language` to Western Arabic (0..9).

Array[Vector3i] **_parse_structured_text**(parser_type: StructuredTextParser, args: Array, text: String) virtual const

Optional.

Default implementation of the BiDi algorithm override function. See StructuredTextParser for more info.

String **_percent_sign**(language: String) virtual const

Optional.

Returns percent sign used in the `language`.

---

bool **_save_support_data**(filename: String) virtual const

Optional.

Saves optional TextServer database (e.g. ICU break iterators and dictionaries) to the file.

---

int **_shaped_get_span_count**(shaped: RID) virtual const

Required.

Returns number of text spans added using _shaped_text_add_string or _shaped_text_add_object.

Variant **_shaped_get_span_meta**(shaped: RID, index: int) virtual const

Required.

Returns text span metadata.

---

void **_shaped_set_span_update_font**(shaped: RID, index: int, fonts: Array[RID], size: int, opentype_features: Dictionary) virtual

Required.

Changes text span font, font size, and OpenType features, without changing the text.

---

bool **_shaped_text_add_object**(shaped: RID, key: Variant, size: Vector2, inline_align: InlineAlignment, length: int, baseline: float) virtual

Required.

Adds inline object to the text buffer, `key` must be unique. In the text, object is represented as `length` object replacement characters.

---

bool **_shaped_text_add_string**(shaped: RID, text: String, fonts: Array[RID], size: int, opentype_features: Dictionary, language: String, meta: Variant) virtual 🔗

Required.

Adds text span and font to draw it to the text buffer.

---

void **_shaped_text_clear**(shaped: RID) virtual 🔗

Required.

Clears text buffer (removes text and inline objects).

---

int **_shaped_text_closest_character_pos**(shaped: RID, pos: int) virtual const 🔗

Optional.

Returns composite character position closest to the `pos`.

---

void **_shaped_text_draw**(shaped: RID, canvas: RID, pos: Vector2, clip_l: float, clip_r: float, color: Color) virtual const 🔗

Optional.

Draw shaped text into a canvas item at a given position, with `color`. `pos` specifies the leftmost point of the baseline (for horizontal layout) or topmost point of the baseline (for vertical layout).

---

void **_shaped_text_draw_outline**(shaped: RID, canvas: RID, pos: Vector2, clip_l: float, clip_r: float, outline_size: int, color: Color) virtual const 🔗

Optional.

Draw the outline of the shaped text into a canvas item at a given position, with `color`. `pos` specifies the leftmost point of the baseline (for horizontal layout) or topmost point of the baseline (for vertical layout).

---

float **_shaped_text_fit_to_width**(shaped: RID, width: float, justification_flags: BitField[JustificationFlag]) virtual

Optional.

Adjusts text width to fit to specified width, returns new text width.

---

float **_shaped_text_get_ascent**(shaped: RID) virtual const

Required.

Returns the text ascent (number of pixels above the baseline for horizontal layout or to the left of baseline for vertical).

---

void **_shaped_text_get_carets**(shaped: RID, position: int, caret: `CaretInfo*`) virtual const

Optional.

Returns shapes of the carets corresponding to the character offset `position` in the text. Returned caret shape is 1 pixel wide rectangle.

PackedInt32Array **_shaped_text_get_character_breaks**(shaped: RID) virtual const

Optional.

Returns array of the composite character boundaries.

---

int **_shaped_text_get_custom_ellipsis**(shaped: RID) virtual const

Optional.

Returns ellipsis character used for text clipping.

String **_shaped_text_get_custom_punctuation**(shaped: RID) virtual const

Optional.

Returns custom punctuation character list, used for word breaking. If set to empty string, server defaults are used.

float **_shaped_text_get_descent**(shaped: RID) virtual const

Required.

Returns the text descent (number of pixels below the baseline for horizontal layout or to the right of baseline for vertical).

Direction **_shaped_text_get_direction**(shaped: RID) virtual const

Optional.

Returns direction of the text.

int **_shaped_text_get_dominant_direction_in_range**(shaped: RID, start: int, end: int) virtual const

Optional.

Returns dominant direction of in the range of text.

int **_shaped_text_get_ellipsis_glyph_count**(shaped: RID) virtual const

Required.

Returns number of glyphs in the ellipsis.

`const Glyph*` **_shaped_text_get_ellipsis_glyphs**(shaped: RID) virtual const

Required.

Returns array of the glyphs in the ellipsis.

---

int **_shaped_text_get_ellipsis_pos**(shaped: RID) virtual const

Required.

Returns position of the ellipsis.

---

int **_shaped_text_get_glyph_count**(shaped: RID) virtual const

Required.

Returns number of glyphs in the buffer.

`const Glyph*` **_shaped_text_get_glyphs**(shaped: RID) virtual const

Required.

Returns an array of glyphs in the visual order.

Vector2 **_shaped_text_get_grapheme_bounds**(shaped: RID, pos: int) virtual const

Optional.

Returns composite character's bounds as offsets from the start of the line.

Direction **_shaped_text_get_inferred_direction**(shaped: RID) virtual const

Optional.

Returns direction of the text, inferred by the BiDi algorithm.

PackedInt32Array **_shaped_text_get_line_breaks**(shaped: RID, width: float, start: int, break_flags: BitField[LineBreakFlag]) virtual const 🔗

Optional.

Breaks text to the lines and returns character ranges for each line.

PackedInt32Array **_shaped_text_get_line_breaks_adv**(shaped: RID, width: PackedFloat32Array, start: int, once: bool, break_flags: BitField[LineBreakFlag]) virtual const 🔗

Optional.

Breaks text to the lines and columns. Returns character ranges for each segment.

---

int **_shaped_text_get_object_glyph**(shaped: RID, key: Variant) virtual const 🔗

Required.

Returns the glyph index of the inline object.

Vector2i **_shaped_text_get_object_range**(shaped: RID, key: Variant) virtual const 🔗

Required.

Returns the character range of the inline object.

Rect2 **_shaped_text_get_object_rect**(shaped: RID, key: Variant) virtual const 🔗

Required.

Returns bounding rectangle of the inline object.

Array **_shaped_text_get_objects**(shaped: RID) virtual const 🔗

Required.

Returns array of inline objects.

Orientation **_shaped_text_get_orientation**(shaped: RID) virtual const 🔗

Optional.

Returns text orientation.

RID **_shaped_text_get_parent**(shaped: RID) virtual const 🔗

Required.

Returns the parent buffer from which the substring originates.

---

bool **_shaped_text_get_preserve_control**(shaped: RID) virtual const 🔗

Optional.

Returns `true` if text buffer is configured to display control characters.

---

bool **_shaped_text_get_preserve_invalid**(shaped: RID) virtual const 🔗

Optional.

Returns `true` if text buffer is configured to display hexadecimal codes in place of invalid characters.

Vector2i **_shaped_text_get_range**(shaped: RID) virtual const 🔗

Required.

Returns substring buffer character range in the parent buffer.

PackedVector2Array **_shaped_text_get_selection**(shaped: RID, start: int, end: int) virtual const

Optional.

Returns selection rectangles for the specified character range.

Vector2 **_shaped_text_get_size**(shaped: RID) virtual const

Required.

Returns size of the text.

---

int **_shaped_text_get_spacing**(shaped: RID, spacing: SpacingType) virtual const

Optional.

Returns extra spacing added between glyphs or lines in pixels.

---

int **_shaped_text_get_trim_pos**(shaped: RID) virtual const

Required.

Returns the position of the overrun trim.

---

float **_shaped_text_get_underline_position**(shaped: RID) virtual const

Required.

Returns pixel offset of the underline below the baseline.

---

float **_shaped_text_get_underline_thickness**(shaped: RID) virtual const

Required.

Returns thickness of the underline.

---

float **_shaped_text_get_width**(shaped: RID) virtual const 🔗

Required.

Returns width (for horizontal layout) or height (for vertical) of the text.

PackedInt32Array **_shaped_text_get_word_breaks**(shaped: RID, grapheme_flags: BitField[GraphemeFlag], skip_grapheme_flags: BitField[GraphemeFlag]) virtual const 🔗

Optional.

Breaks text into words and returns array of character ranges. Use `grapheme_flags` to set what characters are used for breaking (see GraphemeFlag).

---

int **_shaped_text_hit_test_grapheme**(shaped: RID, coord: float) virtual const 🔗

Optional.

Returns grapheme index at the specified pixel offset at the baseline, or `-1` if none is found.

---

int **_shaped_text_hit_test_position**(shaped: RID, coord: float) virtual const 🔗

Optional.

Returns caret character offset at the specified pixel offset at the baseline. This function always returns a valid position.

---

bool **_shaped_text_is_ready**(shaped: RID) virtual const 🔗

Required.

Returns `true` if buffer is successfully shaped.

---

int **_shaped_text_next_character_pos**(shaped: RID, pos: int) virtual const 🔗

Optional.

Returns composite character end position closest to the `pos`.

---

int **_shaped_text_next_grapheme_pos**(shaped: RID, pos: int) virtual const 🔗

Optional.

Returns grapheme end position closest to the `pos`.

---

void **_shaped_text_overrun_trim_to_width**(shaped: RID, width: float, trim_flags: BitField[TextOverrunFlag]) virtual 🔗

Optional.

Trims text if it exceeds the given width.

---

int **_shaped_text_prev_character_pos**(shaped: RID, pos: int) virtual const 🔗

Optional.

Returns composite character start position closest to the `pos`.

---

int **_shaped_text_prev_grapheme_pos**(shaped: RID, pos: int) virtual const 🔗

Optional.

Returns grapheme start position closest to the `pos`.

---

bool **_shaped_text_resize_object**(shaped: RID, key: Variant, size: Vector2, inline_align: InlineAlignment, baseline: float) virtual

Required.

Sets new size and alignment of embedded object.

---

void **_shaped_text_set_bidi_override**(shaped: RID, override: Array) virtual

Optional.

Overrides BiDi for the structured text.

---

void **_shaped_text_set_custom_ellipsis**(shaped: RID, char: int) virtual

Optional.

Sets ellipsis character used for text clipping.

---

void **_shaped_text_set_custom_punctuation**(shaped: RID, punct: String) virtual

Optional.

Sets custom punctuation character list, used for word breaking. If set to empty string, server defaults are used.

---

void **_shaped_text_set_direction**(shaped: RID, direction: Direction) virtual

Optional.

Sets desired text direction. If set to TextServer.DIRECTION_AUTO, direction will be detected based on the buffer contents and current locale.

---

void **_shaped_text_set_orientation**(shaped: RID, orientation: Orientation) virtual 🔗

Optional.

Sets desired text orientation.

void **_shaped_text_set_preserve_control**(shaped: RID, enabled: bool) virtual 🔗

Optional.

If set to `true` text buffer will display control characters.

void **_shaped_text_set_preserve_invalid**(shaped: RID, enabled: bool) virtual 🔗

Optional.

If set to `true` text buffer will display invalid characters as hexadecimal codes, otherwise nothing is displayed.

void **_shaped_text_set_spacing**(shaped: RID, spacing: SpacingType, value: int) virtual 🔗

Optional.

Sets extra spacing added between glyphs or lines in pixels.

bool **_shaped_text_shape**(shaped: RID) virtual 🔗

Required.

Shapes buffer if it's not shaped. Returns `true` if the string is shaped successfully.

`const Glyph*` **_shaped_text_sort_logical**(shaped: RID) virtual 🔗

Required.

Returns text glyphs in the logical order.

RID **_shaped_text_substr**(shaped: RID, start: int, length: int) virtual const

Required.

Returns text buffer for the substring of the text in the `shaped` text buffer (including inline objects).

---

float **_shaped_text_tab_align**(shaped: RID, tab_stops: PackedFloat32Array) virtual

Optional.

Aligns shaped text to the given tab-stops.

---

bool **_shaped_text_update_breaks**(shaped: RID) virtual

Optional.

Updates break points in the shaped text. This method is called by default implementation of text breaking functions.

---

bool **_shaped_text_update_justification_ops**(shaped: RID) virtual

Optional.

Updates justification points in the shaped text. This method is called by default implementation of text justification functions.

---

bool **_spoof_check**(string: String) virtual const

Optional.

Returns `true` if `string` is likely to be an attempt at confusing the reader.

PackedInt32Array **_string_get_character_breaks**(string: String, language: String) virtual const 🔗

Optional.

Returns array of the composite character boundaries.

PackedInt32Array **_string_get_word_breaks**(string: String, language: String, chars_per_line: int) virtual const 🔗

Optional.

Returns an array of the word break boundaries. Elements in the returned array are the offsets of the start and end of words. Therefore the length of the array is always even.

String **_string_to_lower**(string: String, language: String) virtual const 🔗

Optional.

Returns the string converted to lowercase.

String **_string_to_title**(string: String, language: String) virtual const 🔗

Optional.

Returns the string converted to title case.

String **_string_to_upper**(string: String, language: String) virtual const 🔗

Optional.

Returns the string converted to uppercase.

String **_strip_diacritics**(string: String) virtual const 🔗

Optional.

Strips diacritics from the string.

String **_tag_to_name**(tag: int) virtual const

Optional.

Converts OpenType tag to readable feature, variation, script, or language name.

TextServerFallback

Inherits: TextServerExtension **<** TextServer **<** RefCounted **<** Object

A fallback implementation of Godot's text server, without support for BiDi and complex text layout.

Description

A fallback implementation of Godot's text server. This fallback is faster than TextServerAdvanced for processing a lot of text, but it does not support BiDi and complex text layout.

Note: This text server is not part of official Godot binaries. If you want to use it, compile the engine with the option `module_text_server_fb_enabled=yes`.

TextServerManager

Inherits: Object

A singleton for managing TextServer implementations.

Description

TextServerManager is the API backend for loading, enumerating, and switching TextServers.

Note: Switching text server at runtime is possible, but will invalidate all fonts and text buffers. Make sure to unload all controls, fonts, and themes before doing so.

Methods

void	add_interface(interface: TextServer)
TextServer	find_interface(name: String) const
TextServer	get_interface(idx: int) const
int	get_interface_count() const
Array[Dictionary]	get_interfaces() const

TextServer	get_primary_interface() const
void	remove_interface(interface: TextServer)
void	set_primary_interface(index: TextServer)

Signals

interface_added(interface_name: StringName) 🔗

Emitted when a new interface has been added.

interface_removed(interface_name: StringName) 🔗

Emitted when an interface is removed.

Method Descriptions

void **add_interface**(interface: TextServer) 🔗

Registers a TextServer interface.

TextServer **find_interface**(name: String) const 🔗

Finds an interface by its `name`.

TextServer **get_interface**(idx: int) const

Returns the interface registered at a given index.

---

int **get_interface_count**() const

Returns the number of interfaces currently registered.

Array[Dictionary] **get_interfaces**() const

Returns a list of available interfaces, with the index and name of each interface.

TextServer **get_primary_interface**() const

Returns the primary TextServer interface currently in use.

---

void **remove_interface**(interface: TextServer)

Removes an interface. All fonts and shaped text caches should be freed before removing an interface.

---

void **set_primary_interface**(index: TextServer)

Sets the primary TextServer interface.

ThemeDB

Inherits: Object

A singleton that provides access to static information about Theme resources used by the engine and by your project.

Description

This singleton provides access to static information about Theme resources used by the engine and by your projects. You can fetch the default engine theme, as well as your project configured theme.

ThemeDB also contains fallback values for theme properties.

Properties

float	fallback_base_scale	`1.0`
Font	fallback_font	
int	fallback_font_size	`16`
Texture2D	fallback_icon	
StyleBox	fallback_stylebox	

Methods

Theme	get_default_theme()
Theme	get_project_theme()

Signals

fallback_changed()

Emitted when one of the fallback values had been changed. Use it to refresh the look of controls that may rely on the fallback theme items.

Property Descriptions

float **fallback_base_scale** = `1.0`

- void **set_fallback_base_scale**(value: float)
- float **get_fallback_base_scale**()

The fallback base scale factor of every Control node and Theme resource. Used when no other value is available to the control.

See also Theme.default_base_scale.

Font **fallback_font**

- void **set_fallback_font**(value: Font)
- Font **get_fallback_font**()

The fallback font of every Control node and Theme resource. Used when no other value is available to the control.

See also Theme.default_font.

Int **fallback_font_size** = `16`

- void **set_fallback_font_size**(value: int)
- int **get_fallback_font_size**()

The fallback font size of every Control node and Theme resource. Used when no other value is available to the control.

See also Theme.default_font_size.

Texture2D **fallback_icon**

- void **set_fallback_icon**(value: Texture2D)
- Texture2D **get_fallback_icon**()

The fallback icon of every Control node and Theme resource. Used when no other value is available to the control.

StyleBox **fallback_stylebox**

- void **set_fallback_stylebox**(value: StyleBox)
- StyleBox **get_fallback_stylebox**()

The fallback stylebox of every Control node and Theme resource. Used when no other value is available to the control.
